MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this
document by AN974/D

AN974

MC68HC11 Floating-Point Package

INTRODUCTION

The MC68HC11 is a very powerful and capable single-chip microcomputer. Its concise instruction set combined with six powerful addressing modes, true bit manipulation, 16-bit arithmetic operations and a second 16-bit index register make it ideal for control applications requiring both high-speed I/O and high-speed calculations.

While most applications can be implemented by using the 16-bit integer precision of the MC68HC11, certain applications or algorithms may be difficult or impossible to implement without floating-point math. The goal in writing the MC68HC11 floating-point package was to provide a fast, flexible way to do floating-point math for just such applications.

The HC11 floating-point package (HC11FP) implements more than just the four basic math functions (add, subtract, multiply, and divide); it also provides routines to convert from ASCII to floating point and from floating point to ASCII. For those applications that require it, the three basic trig functions SINe, COSine, and TANgent are provided along with some trig utility functions for converting to and from both radians and degrees. The square root function is also included.

For those applications that can benefit by using both integer and floating-point operations, there are routines to convert to and from integer and floating-point format.

The entire floating-point package requires just a little over 2k bytes of memory and only requires ten bytes of page-zero RAM in addition to stack RAM. All temporary variables needed by the floating-point routines, reside on the stack. This feature makes the routines completely reentrant as long as the ten bytes of page zero RAM are saved before using any of the routines. This will allow both interrupt routines and main line programs to use the floating-point package without interfering with one another.

FLOATING-POINT FORMAT

FLOATING-POINT ACCUMULATOR FORMAT

The ten bytes of page-zero RAM are used for two software floating-point accumulators FPACC1 and FPACC2. Each five-byte accumulator consists of a one-byte exponent, a three-byte mantissa, and one byte that is used to indicate the mantissa sign.

The exponent byte is used to indicate the position of the binary point and is biased by decimal 128 ($80) to make floating-point comparisons easier. This one-byte exponent gives a dynamic range of about $1 \times 10 \pm 38$.

The mantissa consists of three bytes (24 bits) and is used to hold both the integer and fractional portion of the floating-point number. The mantissa is always assumed to be "normalized" (i.e., most-significant bit of the most-significant byte a one). A 24-bit mantissa will provide slightly more than seven decimal digits of precision.

A separate byte is used to indicate the sign of the mantissa rather than keeping it in twos complement form so that unsigned arithmetic operations may be used when manipulating the mantissa. A positive mantissa is indicated by this byte being equal to zero ($00). A negative mantissa is indicated by this byte being equal to minus one ($FF).

FPACC1	82 C90FDB 00	+3.1415927
FPACC2	82 C90FDB FF	−3.1415927

MEMORY FORMAT

The way that floating-point numbers are stored in memory or the "memory format" of a floating-point number is slightly different than its floating-point accumulator format. In order to save memory, floating-point numbers are stored in memory in a format called "hidden bit normalized form".

In this format, the number is stored into four consecutive bytes with the exponent residing at the lowest address. The mantissa is stored in the next three consecutive bytes with the most-significant byte stored in the lowest address. Since the most-significant bit of the mantissa in a normalized floating-point number is always a one, this bit can be used to store the sign of the mantissa. This results in positive numbers having the most-significant bit of the mantissa cleared (zero) and negative numbers having their most-significant bit set (one). An example follows:

82 490FDB	+3.1415927
82 C90FDB	−3.1415927

There are four routines that can be used to save and load the floating-point accumulators and at the same time convert between the floating-point accumulator and memory format. These routines are discussed in detail in **FLOATING-POINT ROUTINES**.

ERRORS

There are seven error conditions that may be returned by the HC11 floating-point package. When an error occurs, the condition is indicated to the calling program by setting the carry bit in the condition code register and returning an error code in the A-accumulator. The error codes and their meanings are explained below.

Error #	Meaning
1	Format Error in ASCII to Floating-Point Conversion
2	Floating-Point Overflow
3	Floating-Point Underflow
4	Division by Zero (0)
5	Floating-Point Number too Large or Small to Convert to Integer
6	Square Root of a Negative Number
7	TAN of $\pi/2$ (90°)

NOTE

None of the routines check for valid floating-point numbers in either FPACC1 or FPACC2. Having illegal floating-point values in the floating-point accumulators will produce unpredictable results.

FLOATING-POINT ROUTINES

The following paragraphs provide a description of each routine in the floating-point package. The information provided includes the subroutine name, operation performed, subroutine size, stack space required, other subroutines that are called, input, output, and possible error conditions.

The Stack Space required by the subroutine includes not only that required for the particular routines local variables, but also stack space that is used by any other subroutines that are called including return addresses. Note that the trig functions require a good deal of stack space.

Since some applications may not require all the routines provided in the floating-point package, the description of each routine includes the names of other subroutines that it calls. This makes it easy to determine exactly which subroutines are required for a particular function.

ASCII-TO-FLOATING-POINT CONVERSION

Subroutine Name: ASCFLT
Operation: ASCII (X) ⏵ FPACC1
Size: 352 Bytes (includes NUMERIC subroutine)
Stack Space: 14 Bytes
Calls: NUMERIC, FPNORM, FLTMUL, PSHFPAC2, PULFPAC2
Input: X register points to ASCII string to convert.
Output: FPACC1 contains the floating-point number.
Error Conditions: Floating-point format error may be returned.
Notes: This routine converts an ASCII floating-point number to the format required by all of the floating-point routines. Conversion stops either when a non-decimal character is encountered before the exponent or after one or two exponent digits have been converted. The input format is very flexible. Some examples are shown below.

20.095
0.125
7.2984E + I0
167.824E5
005.9357E − 7
500

FLOATING-POINT MULTIPLY

Subroutine Name: FLTMUL
Operation: FPACC1 × FPACC2 ⏵ FPACC1
Size: 169 Bytes
Stack Space: 10 Bytes
Calls: PSHFPAC2, PULFPAC2, CHCK0
Input: FPACC1 and FPACC2 contain the numbers to be multiplied.
Output: FPACC1 contains the product of the two floating-point accumulators. FPACC2 remains unchanged.
Error Conditions: Overflow, Underflow.

FLOATING-POINT ADD

Subroutine Name:	FLTADD
Operation:	FPACC1 + FPACC2 ⏵ FPACC1
Size:	194 Bytes
Stack Space:	6 Bytes
Calls:	PSHFPAC2, PULFPAC2, CHCK0
Input:	FPACC1 and FPACC2 contain the numbers to be added.
Output:	FPACC1 contains the sum of the two numbers. FPACC2 remains unchanged.
Error Conditions:	Overflow, Underflow.
Notes:	The floating-point add routine performs full signed addition. Both floating-point accumulators may have mantissas with the same or different sign.

FLOATING-POINT SUBTRACT

Subroutine Name:	FLTSUB
Operation:	FPACC1 – FPACC2 ⏵ FPACC1
Size:	12 Bytes
Stack Space:	8 Bytes
Calls:	FLTADD
Input:	FPACC1 and FPACC2 contain the numbers to be subtracted.
Output:	FPACC1 contains the difference of the two numbers (FPACC1-FPACC2). FPACC2 remains unchanged.
Error Conditions:	Overflow, Underflow,
Notes:	Since FLTADD performs full signed addition, the floating-point subtract routine inverts the sign byte of FPACC2, calls FLTADD, and then changes the sign of FPACC2 back to what it was originally.

FLOATING-POINT DIVIDE

Subroutine Name:	FLTDIV
Operation:	FPACC1 ÷ FPACC2 ⏵ FPACC1
Size:	209 Bytes
Stack Space:	11 Bytes
Calls:	PSHFPAC2, PULFPAC2
Input:	FPACC1 and FPACC2 contain the divisor and dividend respectively.
Output:	FPACC1 contains the quotient. FPACC2 remains unchanged.
Error Conditions:	Divide by zero, Overflow, Underflow

FLOATING-POINT-TO-ASCII CONVERSION

Subroutine Name:	FLTASC
Operation:	FPACC1 ⏵ (X)
Size:	370 Bytes
Stack Space:	28 Bytes
Calls:	FLTMUL, FLTCMP, PSHFPAC2, PULFPAC2
Input:	FPACC1 contains the number to be converted to an ASCII string. The index register X points to a 14 byte string buffer.
Output:	The buffer pointed to by the X index register contains an ASCII string that represents the number in FPACC1. The string is terminated with a zero (0) byte and the X register points to the start of the string.
Error Conditions:	None

FLOATING POINT COMPARE

Subroutine Name:	FLTCMP
Operation:	FPACC1 - FPACC2
Size:	42 Bytes
Stack Space:	None
Calls:	None
Input:	FPACC1 and FPACC2 contain the numbers to be compared.
Output:	Condition codes are properly set so that all branch instructions may be used to alter program flow. FPACC1 and FPACC2 remain unchanged.
Error Conditions:	None

UNSIGNED INTEGER TO FLOATING POINT

Subroutine Name:	UINT2FLT
Operation:	(16-bit unsigned integer) ➧ FPACC1
Size:	18 Bytes
Stack Space:	6 Bytes
Calls:	FPNORM, CHCK0
Input:	The lower 16-bits of the FPACC1 mantissa contain an unsigned 16-bit integer.
Output:	FPACC1 contains the floating-point representation of the 16-bit unsigned integer.
Error Conditions:	None

SIGNED INTEGER TO FLOATING POINT

Subroutine Name:	SINT2FLT
Operation:	(16-bit signed integer) ➧ FPACC1
Size:	24 Bytes
Stack Space:	7 Bytes
Calls:	UINT2FLT
Input:	The lower 16-bits of the FPACC1 mantissa contain a signed 16-bit integer.
Output:	FPACC1 contains the floating-point representation of the 16-bit signed integer.
Error Conditions:	None

FLOATING POINT TO INTEGER

Subroutine Name:	FLT2INT
Operation:	FPACC1 ➧ (16-bit signed or unsigned integer)
Size:	74 Bytes
Stack Space:	2 Bytes
Calls:	CHCK0
Input:	FPACC1 may contain a floating-point number in the range $65535 \leq FPACC1 \geq -32767$.
Output:	The lower 16-bits of the FPACC1 mantissa will contain a 16-bit signed or unsigned number.
Error Conditions:	None
Notes:	If the floating-point number in FPACC1 is positive, it will be converted to an unsigned integer. If the number is negative it will be converted to a signed twos complement integer. This type of conversion will allow 16-bit addresses to be represented as positive numbers in floating-point format. Any fractional part of the floating-point number is discarded.

TRANSFER FPACC1 TO FPACC2

Subroutine Name:	TFR1TO2
Operation:	FPACC1 ➧ FPACC2
Size:	13 Bytes
Stack Space:	0 Bytes
Calls:	None
Input:	FPACC1 contains a floating-point number.
Output:	FPACC2 contains the same number as FPACC1.
Error Conditions:	None

FLOATING-POINT FUNCTIONS

The following paragraphs describe the supplied floating-point functions, returned results, and possible error conditions. Note that even though the Taylor series which is used to calculate the trig functions requires that the input angle be expressed in radians; less precision is lost through angle reduction if the angle being reduced is expressed in degrees. Once the angle is reduced, the DEG2RAD subroutine is called to convert the angle to radians.

To reduce the number of factors in the Taylor expansion series all angles are reduced to fall between 0° and 45° by the ANGRED subroutine. This subroutine returns the reduced angle in FPACC1 along with the quad number that the original angle was in, and a flag that tells the calling routine whether it actually needs to calculate the sine or the cosine of the reduced angle to obtain the proper answer.

SQUARE ROOT

Subroutine Name:	FLTSQR
Operation:	$\sqrt{}$ FPACC1 ▸ FPACC1
Size:	104 Bytes
Stack Space:	21 Bytes
Calls:	TFR1TO2, FLTDIV, FLTADD, PSHFPAC2, PULFPAC2
Input:	FPACC1 contains a valid floating-point number.
Output:	FPACC1 contains the square root of the original number. FPACC2 is unchanged.
Error Conditions:	NSQRTERR is returned if the number in FPACC1 is negative and FPACC1 remains unchanged.

SINE

Subroutine Name:	FLTSIN
Operation:	SIN (FPACC1) ▸ FPACC1
Size:	380 Bytes (Includes SINCOS subroutine)
Stack Space:	50 Bytes
Calls:	ANGRED, SINCOS, DEG2RAD, PSHFPAC2, PULFPAC2
Input:	FPACC1 contains an angle in radians in the range $-2\pi \leq$ FPACC1 $\leq +2\pi$.
Output:	FPACC1 contains the sine of FPACC1, and FPACC2 remains unchanged.
Error Conditions:	None
Notes:	The Taylor Expansion Series is used to calculate the sine of the angle between 0° and 45° ($\pi \div 4$). The subroutine ANGRED is called to reduce the input angle to within this range. Spot checks show a maximum error of $+1.5 \times 10^{-7}$ throughout the input range.

COSINE

Subroutine Name:	FLTCOS
Operation:	COS(FPACC1) ▸ FPACC1
Size:	384 Bytes (Includes SINCOS subroutine)
Stack Space:	50 Bytes
Calls:	ANGRED, FLTSIN, DEG2RAD, PSHFPAC2
Input:	FPACC1 contains an angle in radians in the range $-2\pi \leq$ FPACC1 $\leq +2\pi$.
Output:	FPACC1 contains the cosine of FPACC1, and FPACC2 remains unchanged.
Error Conditions:	None
Notes:	The Taylor Expansion Series is used to calculate the cosine of the angle between 0° and 45° ($\pi \div 4$). The subroutine ANGRED is called to reduce the input angle to within this range. Spot checks show a maximum error of $+1.5 \times 10^{-7}$ throughout the input range.

TANGENT

Subroutine Name:	FLTTAN
Operation:	TAN (FPACC1) ⏵ FPACC1
Size:	35 Bytes (Also requires FLTSIN and FLTCOS)
Stack Space:	56 Bytes
Calls:	TFR1TO2, EXG1AND2, FLTSIN, FLTCOS, FLTDIV, PSHFPAC2, PULFPAC2
Input:	FPACC1 contains an angle in radians in the range $-2\pi \leq$ FPACC1 $\leq +2\pi$.
Output:	FPACC1 contains the tangent of the input angle, and FPACC2 remains unchanged.
Error Conditions:	Returns largest legal number if tangent of $\pm\pi \div 2$ is attempted.
Notes:	The tangent of the input angle is calculated by first obtaining the sine and cosine of the input angle and then using the following formula: TAN = SIN ÷ COS. At 89.9° the tangent function is only accurate to 5 decimal digits. For angles greater than 89.9° accuracy decreases rapidly.

DEGREES TO RADIANS CONVERSION

Subroutine Name:	DEG2RAD
Operation:	FPACC1 $\times \pi \div 180$ ⏵ FPACC1
Size:	15 Bytes
Stack Space:	16 Bytes
Calls:	GETFPAC2, FLTMUL
Input:	Any valid floating-point number representing an angle in degrees.
Output:	Input angles equivalent in radians.
Error Conditions:	None

RADIANS TO DEGREES CONVERSION

Subroutine Name:	RAD2DEG
Operation:	FPACC1 $\times 180 \div \pi$ ⏵ FPACC1
Size:	8 Bytes (Also requires DEG2RAD subroutine)
Stack Space:	16 Bytes
Calls:	DEG2RAD
Input:	Any valid floating-point number representing an angle in radians.
Output:	Input angles equivalent in degrees.
Error Conditions:	Overflow, Underflow.

PI

Subroutine Name:	GETPI
Operation:	π ⏵ FPACC1
Size:	6 Bytes
Stack Space:	None
Input:	None
Output:	The value of π is returned in FPACC1.
Error Conditions:	None
Notes:	This routine should be used to obtain the value of π if it is required in calculations since it is accurate to the full 24 bits of the mantissa.

FORMAT CONVERSION ROUTINES

As discussed in **FLOATI9NG-POINT ACCUMULATOR FORMAT** and **MEMORY FORMAT**, the format for floating-point numbers as they appear in the floating-point accumulators is different than the way numbers are stored in memory. This was done primarily to save memory when a large number of floating-point variables are used in a program. Four routines are provided to convert to and from the different formats while at the same time moving a number into or out of the floating-point accumulators. By always using these routines to move numbers into and out of the floating-point accumulators, it would be extremely easy to adapt this floating-point package to work with any other floating-point format.

One example might be to interface this package with code produced by Motorola's 68HC11 'C' compiler. The Motorola 'C' compiler generates code for single-precision floating-point numbers whose internal format is that defined by the *IEEE Standard for Binary Floating-Point Arithmetic*. By rewriting the four routines described below the IEEE format could be easily converted to the format required by this floating-point package.

Get FPACC(x)

Subroutine Name:	GETFPAC1 and GETFPAC2
Operation:	(X) ▸ FPACC1; (X) ▸ FPACC2
Size:	22 Bytes each
Stack Space:	None
Input:	The X index register points to the 'memory formatted' number to be moved into the floating-point accumulator.
Output:	The number pointed to by X is in the specified floating-point accumulator.
Error Conditions:	None

Put FPACC(x)

Subroutine Name:	PUTFPAC1 and PUTFPAC2
Operation:	FPACC1 ▸ (X); FPACC2 ▸ (X)
Size:	22 Bytes each.
Stack Space:	None
Input:	The X index register points to four consecutive memory locations where the number will be stored.
Output:	The floating point accumulator is moved into consecutive memory locations pointed to by the X index register.
Error Conditions:	None

```
0001                    ********************************************************************************
0002                    *                                HC11FP                                        *
0003                    *                                                                              *
0004                    *                             Copyright 1986                                   *
0005                    *                                by                                            *
0006                    *                             Gordon Doughman                                  *
0007                    *                                                                              *
0008                    *  The source code for this floating point package for the MC68HC11            *
0009                    *  may be freely distributed under the rules of public domain. However         *
0010                    *  it is a copyrighted work and as such may not be sold as a product           *
0011                    *  or be included as part of a product for sale without the express            *
0012                    *  permission of the author. Any object code produced by the source            *
0013                    *  code may be included as part of a product for sale.                         *
0014                    *                                                                              *
0015                    *  If there are any questions or comments about the floating point             *
0016                    *  package please feel free to contact me.                                     *
0017                    *                                                                              *
0018                    *                              Gordon Doughman                                 *
0019                    *                              Motorola Semiconductor                          *
0020                    *                              3490 South Dixie Drive                          *
0021                    *                              Dayton, OH  45439                               *
0022                    *                              (513) 294-2231                                  *
0023                    *                                                                              *
0024                    ********************************************************************************
0025                    *
0026                    *
0027                    *
0028 0000                       ORG    $0000
0029                    *
0030 0000               FPACC1EX RMB    1          FLOATING POINT ACCUMULATOR #1..
0031 0001               FPACC1MN RMB    3
0032 0004               MANTSGN1 RMB    1          MANTISSA SIGN FOR FPACC1 (0=+, FF=-).
0033 0005               FPACC2EX RMB    1          FLOATING POINT ACCUMULATOR #2.
0034 0006               FPACC2MN RMB    3
0035 0009               MANTSGN2 RMB    1          MANTISSA SIGN FOR FPACC2 (0=+, FF=-).
0036                    *
0037                    *
0038 0001               FLTFMTER EQU    1         /* floating point format error in ASCFLT */
0039 0002               OVFERR   EQU    2         /* floating point overflow error */
0040 0003               UNFERR   EQU    3         /* floating point underflow error */
0041 0004               DIV0ERR  EQU    4         /* division by 0 error */
0042 0005               TOLGSMER EQU    5         /* number too large or small to convert to int. */
0043 0006               NSQRTERR EQU    6         /* tried to take the square root of negative # */
0044 0007               TAN90ERR EQU    7         /* TANgent of 90 degrees attempted */
0045                    *
0046                    *
```

```
0047                            TTL   ASCFLT
0048                    ********************************************************************************
0049                    *                                                                              *
0050                    *                        ASCII TO FLOATING POINT ROUTINE                       *
0051                    *                                                                              *
0052                    *       This routine will accept most any ASCII floating point format          *
0053                    *       and return a 32-bit floating point number.  The following are          *
0054                    *       some examples of legal ASCII floating point numbers.                   *
0055                    *                                                                              *
0056                    *       20.095                                                                 *
0057                    *       0.125                                                                  *
0058                    *       7.2984E10                                                              *
0059                    *       167.824E5                                                              *
0060                    *       5.9357E-7                                                              *
0061                    *       500                                                                    *
0062                    *                                                                              *
0063                    *       The floating point number returned is in "FPACC1".                     *
0064                    *                                                                              *
0065                    *                                                                              *
0066                    *       The exponent is biased by 128 to facilitate floating point             *
0067                    *       comparisons.  A pointer to the ASCII string is passed to the           *
0068                    *       routine in the D-register.                                             *
0069                    *                                                                              *
0070                    *                                                                              *
0071                    ********************************************************************************
0072                    *
0073                    *
0074                    *       ORG     $0000
0075                    *
0076                    *       FPACC1EX RMB    1           FLOATING POINT ACCUMULATOR #1..
0077                    *       FPACC1MN RMB    3
0078                    *       MANTSGN1 RMB    1           MANTISSA SIGN FOR FPACC1 (0=+, FF=-).
0079                    *       FPACC2EX RMB    1           FLOATING POINT ACCUMULATOR #2.
0080                    *       FPACC2MN RMB    3
0081                    *       MANTSGN2 RMB    1           MANTISSA SIGN FOR FPACC2 (0=+, FF=-).
0082                    *
0083                    *
0084                    *       FLTFMTER EQU    1
0085                    *
0086                    *
0087                    *       LOCAL VARIABLES (ON STACK POINTED TO BY Y)
0088                    *
0089 0000               EXPSIGN EQU   0         EXPONENT SIGN (0=+, FF=-).
0090 0001               PWR10EXP EQU  1         POWER 10 EXPONENT.
0091                    *
0092                    *
0093 C000                       ORG   $C000     (TEST FOR EVB)
0094                    *
0095 C000               ASCFLT  EQU   *
0096 C000 3C                    PSHX            SAVE POINTER TO ASCII STRING.
0097 C001 BD C8 39              JSR   PSHFPAC2  SAVE FPACC2.
0098 C004 CE 00 00              LDX   #0        PUSH ZEROS ON STACK TO INITIALIZE LOCALS.
0099 C007 3C                    PSHX            ALLOCATE 2 BYTES FOR LOCALS.
0100 C008 DF 00                 STX   FPACC1EX  CLEAR FPACC1.
0101 C00A DF 02                 STX   FPACC1EX+2
0102 C00C 7F 00 04              CLR   MANTSGN1  MAKE THE MANTISSA SIGN POSITIVE INITIALLY.
0103 C00F 18 30                 TSY             POINT TO LOCALS.
0104 C011 CD EE 06              LDX   6,Y       GET POINTER TO ASCII STRING.
0105 C014 A6 00         ASCFLT1 LDAA  0,X       GET 1ST CHARACTER IN STRING.
0106 C016 BD C1 55              JSR   NUMERIC   IS IT A NUMBER.
0107 C019 25 28                 BCS   ASCFLT4   YES. GO PROCESS IT.
0108                    *
0109                    *       LEADING MINUS SIGN ENCOUNTERED?
0110                    *
0111 C01B 81 2D         ASCFLT2 CMPA  #'-       NO. IS IT A MINUS SIGN?
```

```
0112 C01D 26 0B                    BNE    ASCFLT3      NO. GO CHECK FOR DECIMAL POINT.
0113 C01F 73 00 04                 COM    MANTSGN1     YES. SET MANTISSA SIGN. LEADING MINUS BEFORE?
0114 C022 08                       INX                 POINT TO NEXT CHARACTER.
0115 C023 A6 00                    LDAA   0,X          GET IT.
0116 C025 BD C1 55                 JSR    NUMERIC      IS IT A NUMBER?
0117 C028 25 19                    BCS    ASCFLT4      YES. GO PROCESS IT.
0118                          *
0119                          *    LEADING DECIMAL POINT?
0120                          *
0121
0122 C02A 81 2E               ASCFLT3 CMPA  #'.          IS IT A DECIMAL POINT?
0123 C02C 26 0B                    BNE    ASCFLT5      NO. FORMAT ERROR.
0124 C02E 08                       INX                 YES. POINT TO NEXT CHARACTER.
0125 C02F A6 00                    LDAA   0,X          GET IT.
0126 C031 BD C1 55                 JSR    NUMERIC      MUST HAVE AT LEAST ONE DIGIT AFTER D.P.
0127 C034 24 03                    BCC    ASCFLT5      GO REPORT ERROR.
0128 C036 7E C0 C1                 JMP    ASCFLT11     GO BUILD FRACTION.
0129                          *
0130                          *    FLOATING POINT FORMAT ERROR
0131                          *
0132 C039 31                  ASCFLT5 INS               DE-ALLOCATE LOCALS.
0133 C03A 31                       INS
0134 C03B BD C8 43                 JSR    PULFPAC2     RESTORE FPACC2.
0135 C03E 38                       PULX                GET POINTER TO TERMINATING CHARACTER IN STRING.
0136 C03F 86 01                    LDAA   #FLTFMTER    FORMAT ERROR.
0137 C041 0D                       SEC                 SET ERROR FLAG.
0138 C042 39                       RTS                 RETURN.
0139                          *
0140                          *    PRE DECIMAL POINT MANTISSA BUILD
0141                          *
0142 C043 A6 00               ASCFLT4 LDAA  0,X
0143 C045 BD C1 55                 JSR    NUMERIC
0144 C048 24 72                    BCC    ASCFLT10
0145 C04A BD C0 D2                 JSR    ADDNXTD
0146 C04D 08                       INX
0147 C04E 24 F3                    BCC    ASCFLT4
0148                          *
0149                          *    PRE DECIMAL POINT MANTISSA OVERFLOW
0150                          *
0151 C050 7C 00 00            ASCFLT6 INC   FPACC1EX     INC FOR EACH DIGIT ENCOUNTERED PRIOR TO D.P.
0152 C053 A6 00                    LDAA   0,X          GET NEXT CHARACTER.
0153 C055 08                       INX                 POINT TO NEXT.
0154 C056 BD C1 55                 JSR    NUMERIC      IS IT S DIGIT?
0155 C059 25 F5                    BCS    ASCFLT6      YES. KEEP BUILDING POWER 10 MANTISSA.
0156 C05B 81 2E                    CMPA   #'.          NO. IS IT A DECIMAL POINT?
0157 C05D 26 0A                    BNE    ASCFLT7      NO. GO CHECK FOR THE EXPONENT.
0158                          *
0159                          *    ANY FRACTIONAL DIGITS ARE NOT SIGNIFIGANT
0160                          *
0161 C05F A6 00               ASCFLT8 LDAA  0,X          GET THE NEXT CHARACTER.
0162 C061 BD C1 55                 JSR    NUMERIC      IS IT A DIGIT?
0163 C064 24 03                    BCC    ASCFLT7      NO. GO CHECK FOR AN EXPONENT.
0164 C066 08                       INX                 POINT TO THE NEXT CHARACTER.
0165 C067 20 F6                    BRA    ASCFLT8      FLUSH REMAINING DIGITS.
0166 C069 81 45               ASCFLT7 CMPA  #'E          NO. IS IT THE EXPONENT?
0167 C06B 27 03                    BEQ    ASCFLT13     YES. GO PROCESS IT.
0168 C06D 7E C1 17                 JMP    FINISH       NO. GO FINISH THE CONVERSION.
0169                          *
0170                          *    PROCESS THE EXPONENT
0171                          *
0172 C070 08                  ASCFLT13 INX              POINT TO NEXT CHARACTER.
0173 C071 A6 00                    LDAA   0,X          GET THE NEXT CHARACTER.
0174 C073 BD C1 55                 JSR    NUMERIC      SEE IF IT'S A DIGIT.
0175 C076 25 15                    BCS    ASCFLT9      YES. GET THE EXPONENT.
0176 C078 81 2D                    CMPA   #'-          NO. IS IT A MINUS SIGN?
0177 C07A 27 06                    BEQ    ASCFLT15     YES. GO FLAG A NEGATIVE EXPONENT.
```

```
0178 C07C 81 2B                    CMPA   #'+          NO. IS IT A PLUS SIGN?
0179 C07E 27 05                    BEQ    ASCFLT16     YES. JUST IGNORE IT.
0180 C080 20 B7                    BRA    ASCFLT5      NO. FORMAT ERROR.
0181 C082 18 63 00          ASCFLT15 COM    EXPSIGN,Y    FLAG A NEGATIVE EXPONENT. IS IT 1ST?
0182 C085 08                ASCFLT16 INX                 POINT TO NEXT CHARACTER.
0183 C086 A6 00                    LDAA   0,X          GET NEXT CHARACTER.
0184 C088 BD C1 55                 JSR    NUMERIC      IS IT A NUMBER?
0185 C08B 24 AC                    BCC    ASCFLT5      NO. FORMAT ERROR.
0186 C08D 80 30             ASCFLT9  SUBA   #$30         MAKE IT BINARY.
0187 C08F 18 A7 01                 STAA   PWR10EXP,Y   BUILD THE POWER 10 EXPONENT.
0188 C092 08                       INX                 POINT TO NEXT CHARACTER.
0189 C093 A6 00                    LDAA   0,X          GET IT.
0190 C095 BD C1 55                 JSR    NUMERIC      IS IT NUMERIC?
0191 C098 24 13                    BCC    ASCFLT14     NO. GO FINISH UP THE CONVERSION.
0192 C09A 18 E6 01                 LDAB   PWR10EXP,Y   YES. GET PREVIOUS DIGIT.
0193 C09D 58                       LSLB                MULT. BY 2.
0194 C09E 58                       LSLB                NOW BY 4.
0195 C09F 18 EB 01                 ADDB   PWR10EXP,Y   BY 5.
0196 C0A2 58                       LSLB                BY 10.
0197 C0A3 80 30                    SUBA   #$30         MAKE SECOND DIGIT BINARY.
0198 C0A5 1B                       ABA                 ADD IT TO FIRST DIGIT.
0199 C0A6 18 A7 01                 STAA   PWR10EXP,Y
0200 C0A9 81 26                    CMPA   #38          IS THE EXPONENT OUT OF RANGE?
0201 C0AB 22 8C                    BHI    ASCFLT5      YES. REPORT ERROR.
0202 C0AD 18 A6 01          ASCFLT14 LDAA   PWR10EXP,Y   GET POWER 10 EXPONENT.
0203 C0B0 18 6D 00                 TST    EXPSIGN,Y    WAS IT NEGATIVE?
0204 C0B3 2A 01                    BPL    ASCFLT12     NO. GO ADD IT TO BUILT 10 PWR EXPONENT.
0205 C0B5 40                       NEGA
0206 C0B6 9B 00             ASCFLT12 ADDA   FPACC1EX     FINAL TOTAL PWR 10 EXPONENT.
0207 C0B8 97 00                    STAA   FPACC1EX     SAVE RESULT.
0208 C0BA 20 5B                    BRA    FINISH       GO FINISH UP CONVERSION.
0209                        *
0210                        *      PRE-DECIMAL POINT NON-DIGIT FOUND, IS IT A DECIMAL POINT?
0211                        *
0212 C0BC 81 2E             ASCFLT10 CMPA  #'.          IS IT A DECIMAL POINT?
0213 C0BE 26 A9                    BNE    ASCFLT7      NO. GO CHECK FOR THE EXPONENT.
0214 C0C0 08                       INX                 YES. POINT TO NEXT CHARACTER.
0215                        *
0216                        *      POST DECIMAL POINT PROCESSING
0217                        *
0218 C0C1 A6 00             ASCFLT11 LDAA  0,X          GET NEXT CHARACTER.
0219 C0C3 BD C1 55                 JSR    NUMERIC      IS IT NUMERIC?
0220 C0C6 24 A1                    BCC    ASCFLT7      NO. GO CHECK FOR EXPONENT.
0221 C0C8 8D 08                    BSR    ADDNXTD      YES. ADD IN THE DIGIT.
0222 C0CA 08                       INX                 POINT TO THE NEXT CHARACTER.
0223 C0CB 25 92                    BCS    ASCFLT8      IF OVER FLOW, FLUSH REMAINING DIGITS.
0224 C0CD 7A 00 00                 DEC    FPACC1EX     ADJUST THE 10 POWER EXPONENT.
0225 C0D0 20 EF                    BRA    ASCFLT11     PROCESS ALL FRACTIONAL DIGITS.
0226                        *
0227                        *
0228                        *
0229 C0D2 96 01             ADDNXTD  LDAA  FPACC1MN     GET UPPER 8 BITS.
0230 C0D4 97 06                    STAA   FPACC2MN     COPY INTO FPAC2.
0231 C0D6 DC 02                    LDD    FPACC1MN+1   GET LOWER 16 BITS OF MANTISSA.
0232 C0D8 DD 07                    STD    FPACC2MN+1   COPY INTO FPACC2.
0233 C0DA 05                       LSLD                MULT. BY 2.
0234 C0DB 79 00 01                 ROL    FPACC1MN     OVERFLOW?
0235 C0DE 25 2E                    BCS    ADDNXTD1     YES. DON'T ADD THE DIGIT IN.
0236 C0E0 05                       LSLD                MULT BY 4.
0237 C0E1 79 00 01                 ROL    FPACC1MN     OVERFLOW?
0238 C0E4 25 28                    BCS    ADDNXTD1     YES. DON'T ADD THE DIGIT IN.
0239 C0E6 D3 07                    ADDD   FPACC2MN+1   BY 5.
0240 C0E8 36                       PSHA                SAVE A.
0241 C0E9 96 01                    LDAA   FPACC1MN     GET UPPER 8 BITS.
0242 C0EB 89 00                    ADCA   #0           ADDIN POSSABLE CARRY FROM LOWER 16 BITS.
0243 C0ED 9B 06                    ADDA   FPACC2MN     ADD IN UPPER 8 BITS.
```

```
0244 C0EF 97 01                    STAA   FPACC1MN     SAVE IT.
0245 C0F1 32                       PULA                RESTORE A.
0246 C0F2 25 1A                    BCS    ADDNXTD1     OVERFLOW? IF SO DON'T ADD IT IN.
0247 C0F4 05                       LSLD                BY 10.
0248 C0F5 79 00 01                 ROL    FPACC1MN
0249 C0F8 DD 02                    STD    FPACC1MN+1   SAVE THE LOWER 16 BITS.
0250 C0FA 25 12                    BCS    ADDNXTD1     OVERFLOW? IF SO DON'T ADD IT IN.
0251 C0FC E6 00                    LDAB   0,X          GET CURRENT DIGIT.
0252 C0FE C0 30                    SUBB   #$30         MAKE IT BINARY.
0253 C100 4F                       CLRA                16-BIT.
0254 C101 D3 02                    ADDD   FPACC1MN+1   ADD IT IN TO TOTAL.
0255 C103 DD 02                    STD    FPACC1MN+1   SAVE THE RESULT.
0256 C105 96 01                    LDAA   FPACC1MN     GET UPPER 8 BITS.
0257 C107 89 00                    ADCA   #0           ADD IN POSSIBLE CARRY. OVERFLOW?
0258 C109 25 03                    BCS    ADDNXTD1     YES. COPY OLD MANTISSA FROM FPACC2.
0259 C10B 97 01                    STAA   FPACC1MN     NO. EVERYHING OK.
0260 C10D 39                       RTS                 RETURN.
0261 C10E DC 07            ADDNXTD1 LDD    FPACC2MN+1  RESTORE THE ORIGINAL MANTISSA BECAUSE
0262 C110 DD 02                    STD    FPACC1MN+1   OF OVERFLOW.
0263 C112 96 06                    LDAA   FPACC2MN
0264 C114 97 01                    STAA   FPACC1MN
0265 C116 39                       RTS                 RETURN.
0266                       *
0267                       *
0268                       *
0269                       *       NOW FINISH UP CONVERSION BY MULTIPLYING THE RESULTANT MANTISSA
0270                       *       BY 10 FOR EACH POSITIVE POWER OF 10 EXPONENT RECIEVED OR BY .1
0271                       *       (DIVIDE BY 10) FOR EACH NEGATIVE POWER OF 10 EXPONENT RECIEVED.
0272                       *
0273                       *
0274 C117                  FINISH  EQU    *
0275 C117 CD EF 06                 STX    6,Y          SAVE POINTER TO TERMINATING CHARACTER IN STRING.
0276 C11A CE 00 00                 LDX    #FPACC1EX    POINT TO FPACC1.
0277 C11D BD C1 80                 JSR    CHCK0        SEE IF THE NUMBER IS ZERO.
0278 C120 27 2C                    BEQ    FINISH3      QUIT IF IT IS.
0279 C122 96 00                    LDAA   FPACC1EX     GET THE POWER 10 EXPONENT.
0280 C124 18 A7 01                 STAA   PWR10EXP,Y   SAVE IT.
0281 C127 86 98                    LDAA   #$80+24      SET UP INITIAL EXPONENT (# OF BITS + BIAS).
0282 C129 97 00                    STAA   FPACC1EX
0283 C12B BD C1 61                 JSR    FPNORM       GO NORMALIZE THE MANTISSA.
0284 C12E 18 6D 01                 TST    PWR10EXP,Y   IS THE POWER 10 EXPONENT POSITIVE OR ZERO?
0285 C131 27 1B                    BEQ    FINISH3      IT'S ZERO, WE'RE DONE.
0286 C133 2A 0B                    BPL    FINISH1      IT'S POSITIVE MULTIPLY BY 10.
0287 C135 CE C1 8B                 LDX    #CONSTP1     NO. GET CONSTANT .1 (DIVIDE BY 10).
0288 C138 BD C8 66                 JSR    GETFPAC2     GET CONSTANT INTO FPACC2.
0289 C13B 18 60 01                 NEG    PWR10EXP,Y   MAKE THE POWER 10 EXPONENT POSITIVE.
0290 C13E 20 06                    BRA    FINISH2      GO DO THE MULTIPLIES.
0291 C140 CE C1 8F         FINISH1 LDX    #CONST10     GET CONSTANT '10' TO MULTIPLY BY.
0292 C143 BD C8 66                 JSR    GETFPAC2     GET CONSTANT INTO FPACC2.
0293 C146 BD C1 93         FINISH2 JSR    FLTMUL       GO MULTIPLY FPACC1 BY FPACC2, RESULT IN FPACC1.
0294 C149 18 6A 01                 DEC    PWR10EXP,Y   DECREMENT THE POWER 10 EXPONENT.
0295 C14C 26 F8                    BNE    FINISH2      GO CHECK TO SEE IF WE'RE DONE.
0296 C14E 31               FINISH3 INS                 DE-ALLOCATE LOCALS.
0297 C14F 31                       INS
0298 C150 BD C8 43                 JSR    PULFPAC2     RESTORE FPACC2.
0299 C153 38                       PULX                GET POINTER TO TERMINATING CHARACTER IN STRING.
0300 C154 39                       RTS                 RETURN WITH NUMBER IN FPACC1.
0301                       *
0302                       *
0303 C155                  NUMERIC EQU    *
0304 C155 81 30                    CMPA   #'0          IS IT LESS THAN AN ASCII 0?
0305 C157 25 06                    BLO    NUMERIC1     YES. NOT NUMERIC.
0306 C159 81 39                    CMPA   #'9          IS IT GREATER THAN AN ASCII 9?
0307 C15B 22 02                    BHI    NUMERIC1     YES. NOT NUMERIC.
0308 C15D 0D                       SEC                 IT WAS NUMERIC. SET THE CARRY.
0309 C15E 39                       RTS                 RETURN.
```

```
0310 C15F 0C              NUMERIC1 CLC                NON-NUMERIC CHARACTER. CLEAR THE CARRY.
0311 C160 39                       RTS                RETURN.
0312                      *
0313 C161                 FPNORM   EQU   *
0314 C161 CE 00 00                 LDX   #FPACC1EX    POINT TO FPACC1.
0315 C164 8D 1A                    BSR   CHCK0        CHECK TO SEE IF IT'S 0.
0316 C166 27 14                    BEQ   FPNORM3      YES. JUST RETURN.
0317 C168 7D 00 01                 TST   FPACC1MN     IS THE NUMBER ALREADY NORMALIZED?
0318 C16B 2B 0F                    BMI   FPNORM3      YES. JUST RETURN..
0319 C16D DC 02           FPNORM1  LDD   FPACC1MN+1   GET THE LOWER 16 BITS OF THE MANTISSA.
0320 C16F 7A 00 00        FPNORM2  DEC   FPACC1EX     DECREMENT THE EXPONENT FOR EACH SHIFT.
0321 C172 27 0A                    BEQ   FPNORM4      EXPONENT WENT TO 0. UNDERFLOW.
0322 C174 05                       LSLD               SHIFT THE LOWER 16 BITS.
0323 C175 79 00 01                 ROL   FPACC1MN     ROTATE THE UPPER 8 BITS. NUMBER NORMALIZED?
0324 C178 2A F5                    BPL   FPNORM2      NO. KEEP SHIFTING TO THE LEFT.
0325 C17A DD 02                    STD   FPACC1MN+1   PUT THE LOWER 16 BITS BACK INTO FPACC1.
0326 C17C 0C              FPNORM3  CLC                SHOW NO ERRORS.
0327 C17D 39                       RTS                YES. RETURN.
0328 C17E 0D              FPNORM4  SEC                FLAG ERROR.
0329 C17F 39                       RTS                RETURN.
0330                      *
0331 C180                 CHCK0    EQU   *           CHECKS FOR ZERO IN FPACC POINTED TO BY X.
0332 C180 37                       PSHB               SAVE D.
0333 C181 36                       PSHA
0334 C182 EC 00                    LDD   0,X          GET FPACC EXPONENT & HIGH 8 BITS.
0335 C184 26 02                    BNE   CHCK01       NOT ZERO. RETURN.
0336 C186 EC 02                    LDD   2,X          CHECK LOWER 16 BITS.
0337 C188 32              CHCK01   PULA               RESTORE D.
0338 C189 33                       PULB
0339 C18A 39                       RTS                RETURN WITH CC SET.
0340                      *
0341 C18B 7D 4C CC CD     CONSTP1  FCB   $7D,$4C,$CC,$CD      0.1 DECIMAL
0342 C18F 84 20 00 00     CONST10  FCB   $84,$20,$00,$00      10.0 DECIMAL
0343                      *
0344                      *
```

```
                            TTL    FLTMUL
                    ********************************************************************************
                    *                                                                              *
                    *                       FPMULT: FLOATING POINT MULTIPLY                        *
                    *                                                                              *
                    *       THIS FLOATING POINT MULTIPLY ROUTINE MULTIPLIES "FPACC1" BY            *
                    *       "FPACC2" AND PLACES THE RESULT IN TO FPACC1. FPACC2 REMAINS            *
                    *       UNCHANGED.                                                             *
                    *                          WORSE CASE = 2319 CYCLES = 1159 uS @ 2MHz           *
                    *                                                                              *
                    ********************************************************************************
                    *
                    *
C193                FLTMUL  EQU    *
C193 BD C8 39               JSR    PSHFPAC2     SAVE FPACC2.
C196 CE 00 00               LDX    #FPACC1EX    POINT TO FPACC1
C199 BD C1 80               JSR    CHCK0        CHECK TO SEE IF FPACC1 IS ZERO.
C19C 27 31                  BEQ    FPMULT3      IT IS. ANSWER IS 0.
C19E CE 00 05               LDX    #FPACC2EX    POINT TO FPACC2.
C1A1 BD C1 80               JSR    CHCK0        IS IT 0?
C1A4 26 08                  BNE    FPMULT4      NO. CONTINUE.
C1A6 4F                     CLRA                CLEAR D.
C1A7 5F                     CLRB
C1A8 DD 00                  STD    FPACC1EX     MAKE FPACC1 0.
C1AA DD 02                  STD    FPACC1MN+1
C1AC 20 21                  BRA    FPMULT3      RETURN.
C1AE 96 04          FPMULT4 LDAA   MANTSGN1     GET FPACC1 EXPONENT.
C1B0 98 09                  EORA   MANTSGN2     SET THE SIGN OF THE RESULT.
C1B2 97 04                  STAA   MANTSGN1     SAVE THE SIGN OF THE RESULT.
C1B4 96 00                  LDAA   FPACC1EX     GET FPACC1 EXPONENT.
C1B6 9B 05                  ADDA   FPACC2EX     ADD IT TO FPACC2 EXPONENT.
C1B8 2A 07                  BPL    FPMULT1      IF RESULT IS MINUS AND
C1BA 24 0C                  BCC    FPMULT2      THE CARRY IS SET THEN:
C1BC 86 02          FPMULT5 LDAA   #OVFERR      OVERFLOW ERROR.
C1BE 0D                     SEC                 SET ERROR FLAG.
C1BF 20 14                  BRA    FPMULT6      RETURN.
C1C1 25 05          FPMULT1 BCS    FPMULT2      IF RESULT IS PLUS & THE CARRY IS SET THEN ALL OK.
C1C3 86 03                  LDAA   #UNFERR      ELSE UNDERFLOW ERROR OCCURED.
C1C5 0D                     SEC                 FLAG ERROR.
C1C6 20 0D                  BRA    FPMULT6      RETURN.
C1C8 8B 80          FPMULT2 ADDA   #$80         ADD 128 BIAS BACK IN THAT WE LOST.
C1CA 97 00                  STAA   FPACC1EX     SAVE THE NEW EXPONENT.
C1CC BD C1 D9               JSR    UMULT        GO MULTIPLY THE "INTEGER" MANTISSAS.
C1CF 7D 00 00       FPMULT3 TST    FPACC1EX     WAS THERE AN OVERFLOW ERROR FROM ROUNDING?
C1D2 27 E8                  BEQ    FPMULT5      YES. RETURN ERROR.
C1D4 0C                     CLC                 SHOW NO ERRORS.
C1D5 BD C8 43       FPMULT6 JSR    PULFPAC2     RESTORE FPACC2.
C1D8 39                     RTS
                    *
                    *
C1D9                UMULT   EQU    *
C1D9 CE 00 00               LDX    #0
C1DC 3C                     PSHX                CREATE PARTIAL PRODUCT REGISTER AND COUNTER.
C1DD 3C                     PSHX
C1DE 30                     TSX                 POINT TO THE VARIABLES.
C1DF 86 18                  LDAA   #24          SET COUNT TO THE NUMBER OF BITS.
C1E1 A7 00                  STAA   0,X
C1E3 96 08          UMULT1  LDAA   FPACC2MN+2   GET THE L.S. BYTE OF THE MULTIPLIER.
C1E5 44                     LSRA                PUT L.S. BIT IN CARRY.
C1E6 24 0C                  BCC    UMULT2       IF CARRY CLEAR, DON'T ADD MULTIPLICAND TO P.P.
C1E8 DC 02                  LDD    FPACC1MN+1   GET MULTIPLICAND L.S. 16 BITS.
C1EA E3 02                  ADDD   2,X          ADD TO PARTIAL PRODUCT.
C1EC ED 02                  STD    2,X          SAVE IN P.P.
C1EE 96 01                  LDAA   FPACC1MN     GET UPPER 8 BITS OF MULTIPLICAND.
C1F0 A9 01                  ADCA   1,X          ADD IT W/ CARRY TO P.P.
```

```
0410 C1F2 A7 01                    STAA   1,X          SAVE TO PARTIAL PRODUCT.
0411 C1F4 66 01             UMULT2 ROR    1,X          ROTATE PARTIAL PRODUCT TO THE RIGHT.
0412 C1F6 66 02                    ROR    2,X
0413 C1F8 66 03                    ROR    3,X
0414 C1FA 76 00 06                 ROR    FPACC2MN     SHIFT THE MULTIPLIER TO THE RIGHT 1 BIT.
0415 C1FD 76 00 07                 ROR    FPACC2MN+1
0416 C200 76 00 08                 ROR    FPACC2MN+2
0417 C203 6A 00                    DEC    0,X          DONE YET?
0418 C205 26 DC                    BNE    UMULT1       NO. KEEP GOING.
0419 C207 6D 01                    TST    1,X          DOES PARTIAL PRODUCT NEED TO BE NORMALIZED?
0420 C209 2B 0C                    BMI    UMULT3       NO. GET ANSWER & RETURN.
0421 C20B 78 00 06                 LSL    FPACC2MN     GET BIT THAT WAS SHIFTED OUT OF P.P REGISTER.
0422 C20E 69 03                    ROL    3,X          PUT IT BACK INTO THE PARTIAL PRODUCT.
0423 C210 69 02                    ROL    2,X
0424 C212 69 01                    ROL    1,X
0425 C214 7A 00 00                 DEC    FPACC1EX     FIX EXPONENT.
0426 C217 7D 00 06          UMULT3 TST    FPACC2MN     DO WE NEED TO ROUND THE PARTIAL PRODUCT?
0427 C21A 2A 18                    BPL    UMULT4       NO. JUST RETURN.
0428 C21C EC 02                    LDD    2,X          YES. GET THE LEAST SIGNIFIGANT 16 BITS.
0429 C21E C3 00 01                 ADDD   #1           ADD 1.
0430 C221 ED 02                    STD    2,X          SAVE RESULT.
0431 C223 A6 01                    LDAA   1,X          PROPIGATE THROUGH.
0432 C225 89 00                    ADCA   #0
0433 C227 A7 01                    STAA   1,X
0434 C229 24 09                    BCC    UMULT4       IF CARRY CLEAR ALL IS OK.
0435 C22B 66 01                    ROR    1,X          IF NOT OVERFLOW. ROTATE CARRY INTO P.P.
0436 C22D 66 02                    ROR    2,X
0437 C22F 66 03                    ROR    3,X
0438 C231 7C 00 00                 INC    FPACC1EX     UP THE EXPONENT.
0439 C234 31                UMULT4 INS                 TAKE COUNTER OFF STACK.
0440 C235 38                       PULX                GET M.S. 16 BITS OF PARTIAL PRODUCT.
0441 C236 DF 01                    STX    FPACC1MN     PUT IT IN FPACC1.
0442 C238 32                       PULA                GET L.S. 8 BITS OF PARTIAL PRODUCT.
0443 C239 97 03                    STAA   FPACC1MN+2   PUT IT IN FPACC1.
0444 C23B 39                       RTS                 RETURN.
0445                        *
0446                        *
0447                        *
```

```
0448                                TTL   FLTADD
0449                        ********************************************************************************
0450                        *                                                                              *
0451                        *                        FLOATING POINT ADDITION                               *
0452                        *                                                                              *
0453                        *       This subroutine performs floating point addition of the two numbers    *
0454                        *       in FPACC1 and FPACC2.  The result of the addition is placed in         *
0455                        *       FPACC1 while FPACC2 remains unchanged.  This subroutine performs       *
0456                        *       full signed addition so either number may be of the same or opposite   *
0457                        *       sign.                                                                  *
0458                        *                              WORSE CASE = 1030 CYCLES = 515 uS @ 2MHz        *
0459                        *                                                                              *
0460                        ********************************************************************************
0461                        *
0462                        *
0463 C23C                  FLTADD   EQU   *
0464 C23C BD C8 39                  JSR   PSHFPAC2    SAVE FPACC2.
0465 C23F CE 00 05                  LDX   #FPACC2EX   POINT TO FPACC2
0466 C242 BD C1 80                  JSR   CHCK0       IS IT ZERO?
0467 C245 26 05                     BNE   FLTADD1     NO. GO CHECK FOR 0 IN FPACC1.
0468 C247 0C               FLTADD6  CLC               NO ERRORS.
0469 C248 BD C8 43         FLTADD10 JSR   PULFPAC2    RESTORE FPACC2.
0470 C24B 39                        RTS               ANSWER IN FPACC1. RETURN.
0471 C24C CE 00 00         FLTADD1  LDX   #FPACC1EX   POINT TO FPACC1.
0472 C24F BD C1 80                  JSR   CHCK0       IS IT ZERO?
0473 C252 26 0E                     BNE   FLTADD2     NO. GO ADD THE NUMBER.
0474 C254 DC 05            FLTADD4  LDD   FPACC2EX    ANSWER IS IN FPACC2. MOVE IT INTO FPACC1.
0475 C256 DD 00                     STD   FPACC1EX
0476 C258 DC 07                     LDD   FPACC2MN+1  MOVE LOWER 16 BITS OF MANTISSA.
0477 C25A DD 02                     STD   FPACC1MN+1
0478 C25C 96 09                     LDAA  MANTSGN2    MOVE FPACC2 MANTISSA SIGN INTO FPACC1.
0479 C25E 97 04                     STAA  MANTSGN1
0480 C260 20 E5                     BRA   FLTADD6     RETURN.
0481 C262 96 00            FLTADD2  LDAA  FPACC1EX    GET FPACC1 EXPONENT.
0482 C264 91 05                     CMPA  FPACC2EX    ARE THE EXPONENTS THE SAME?
0483 C266 27 23                     BEQ   FLTADD7     YES. GO ADD THE MANTISSA'S.
0484 C268 90 05                     SUBA  FPACC2EX    NO. FPACC1EX-FPACC2EX. IS FPACC1 > FPACC2?
0485 C26A 2A 0F                     BPL   FLTADD3     YES. GO CHECK RANGE.
0486 C26C 40                        NEGA              NO. FPACC1 < FPACC2. MAKE DIFFERENCE POSITIVE.
0487 C26D 81 17                     CMPA  #23         ARE THE NUMBERS WITHIN RANGE?
0488 C26F 22 E3                     BHI   FLTADD4     NO. FPACC2 IS LARGER. GO MOVE IT INTO FPACC1.
0489 C271 16                        TAB               PUT DIFFERENCE IN B.
0490 C272 DB 00                     ADDB  FPACC1EX    CORRECT FPACC1 EXPONENT.
0491 C274 D7 00                     STAB  FPACC1EX    SAVE THE RESULT.
0492 C276 CE 00 01                  LDX   #FPACC1MN   POINT TO FPACC1 MANTISSA.
0493 C279 20 07                     BRA   FLTADD5     GO DENORMALIZE FPACC1 FOR THE ADD.
0494 C27B 81 17            FLTADD3  CMPA  #23         FPACC1 > FPACC2. ARE THE NUMBERS WITHIN RANGE?
0495 C27D 22 C8                     BHI   FLTADD6     NO. ANSWER ALREADY IN FPACC1. JUST RETURN.
0496 C27F CE 00 06                  LDX   #FPACC2MN   POINT TO THE MANTISSA TO DENORMALIZE.
0497 C282 64 00            FLTADD5  LSR   0,X         SHIFT THE FIRST BYTE OF THE MANTISSA.
0498 C284 66 01                     ROR   1,X         THE SECOND.
0499 C286 66 02                     ROR   2,X         AND THE THIRD.
0500 C288 4A                        DECA              DONE YET?
0501 C289 26 F7                     BNE   FLTADD5     NO. KEEP SHIFTING.
0502 C28B 96 04            FLTADD7  LDAA  MANTSGN1    GET FPACC1 MANTISSA SIGN.
0503 C28D 91 09                     CMPA  MANTSGN2    ARE THE SIGNS THE SAME?
0504 C28F 27 4B                     BEQ   FLTADD11    YES. JUST GO ADD THE TWO MANTISSAS.
0505 C291 7D 00 04                  TST   MANTSGN1    NO. IS FPACC1 THE NEGATIVE NUMBER?
0506 C294 2A 14                     BPL   FLTADD8     NO. GO DO FPACC1-FPACC2.
0507 C296 DE 06                     LDX   FPACC2MN    YES. EXCHANGE FPACC1 & FPACC2 BEFORE THE SUB.
0508 C298 3C                        PSHX              SAVE IT.
0509 C299 DE 01                     LDX   FPACC1MN    GET PART OF FPACC1.
0510 C29B DF 06                     STX   FPACC2MN    PUT IT IN FPACC2.
0511 C29D 38                        PULX              GET SAVED PORTION OF FPACC2
0512 C29E DF 01                     STX   FPACC1MN    PUT IT IN FPACC1.
```

```
0513 C2A0 DE 08                    LDX    FPACC2MN+2   GET LOWER 8 BITS & SIGN OF FPACC2.
0514 C2A2 3C                       PSHX                SAVE IT.
0515 C2A3 DE 03                    LDX    FPACC1MN+2   GET LOWER 8 BITS & SIGN OF FPACC1.
0516 C2A5 DF 08                    STX    FPACC2MN+2   PUT IT IN FPACC2.
0517 C2A7 38                       PULX                GET SAVED PART OF FPACC2.
0518 C2A8 DF 03                    STX    FPACC1MN+2   PUT IT IN FPACC1.
0519 C2AA DC 02          FLTADD8   LDD    FPACC1MN+1   GET LOWER 16 BITS OF FPACC1.
0520 C2AC 93 07                    SUBD   FPACC2MN+1   SUBTRACT LOWER 16 BITS OF FPACC2.
0521 C2AE DD 02                    STD    FPACC1MN+1   SAVE RESULT.
0522 C2B0 96 01                    LDAA   FPACC1MN     GET HIGH 8 BITS OF FPACC1 MANTISSA.
0523 C2B2 92 06                    SBCA   FPACC2MN     SUBTRACT HIGH 8 BITS OF FPACC2.
0524 C2B4 97 01                    STAA   FPACC1MN     SAVE THE RESULT. IS THE RESULT NEGATIVE?
0525 C2B6 24 16                    BCC    FLTADD9      NO. GO NORMALIZE THE RESULT.
0526 C2B8 96 01                    LDAA   FPACC1MN     YES. NEGATE THE MANTISSA.
0527 C2BA 43                       COMA
0528 C2BB 36                       PSHA                SAVE THE RESULT.
0529 C2BC DC 02                    LDD    FPACC1MN+1   GET LOWER 16 BITS.
0530 C2BE 53                       COMB                FORM THE ONE'S COMPLEMENT.
0531 C2BF 43                       COMA
0532 C2C0 C3 00 01                 ADDD   #1           FORM THE TWO'S COMPLEMENT.
0533 C2C3 DD 02                    STD    FPACC1MN+1   SAVE THE RESULT.
0534 C2C5 32                       PULA                GET UPPER 8 BITS BACK.
0535 C2C6 89 00                    ADCA   #0           ADD IN POSSIBLE CARRY.
0536 C2C8 97 01                    STAA   FPACC1MN     SAVE RESULT.
0537 C2CA 86 FF                    LDAA   #$FF         SHOW THAT FPACC1 IS NEGATIVE.
0538 C2CC 97 04                    STAA   MANTSGN1
0539 C2CE BD C1 61       FLTADD9   JSR    FPNORM       GO NORMALIZE THE RESULT.
0540 C2D1 24 06                    BCC    FLTADD12     EVERYTHING'S OK SO RETURN.
0541 C2D3 86 03                    LDAA   #UNFERR      UNDERFLOW OCCURED DURING NORMALIZATION.
0542 C2D5 0D                       SEC                 FLAG ERROR.
0543 C2D6 7E C2 48                 JMP    FLTADD10     RETURN.
0544 C2D9 7E C2 47       FLTADD12  JMP    FLTADD6      CAN'T BRANCH THAT FAR FROM HERE.
0545                     *
0546 C2DC DC 02          FLTADD11  LDD    FPACC1MN+1   GET LOWER 16 BITS OF FPACC1.
0547 C2DE D3 07                    ADDD   FPACC2MN+1   ADD IT TO THE LOWER 16 BITS OF FPACC2.
0548 C2E0 DD 02                    STD    FPACC1MN+1   SAVE RESULT IN FPACC1.
0549 C2E2 96 01                    LDAA   FPACC1MN     GET UPPER 8 BITS OF FPACC1.
0550 C2E4 99 06                    ADCA   FPACC2MN     ADD IT (WITH CARRY) TO UPPER 8 BITS OF FPACC2.
0551 C2E6 97 01                    STAA   FPACC1MN     SAVE THE RESULT.
0552 C2E8 24 EF                    BCC    FLTADD12     NO OVERFLOW SO JUST RETURN.
0553 C2EA 76 00 01                 ROR    FPACC1MN     PUT THE CARRY INTO THE MANTISSA.
0554 C2ED 76 00 02                 ROR    FPACC1MN+1   PROPIGATE THROUGH MANTISSA.
0555 C2F0 76 00 03                 ROR    FPACC1MN+2
0556 C2F3 7C 00 00                 INC    FPACC1EX     UP THE MANTISSA BY 1.
0557 C2F6 26 E1                    BNE    FLTADD12     EVERYTHING'S OK JUST RETURN.
0558 C2F8 86 02                    LDAA   #OVFERR      RESULT WAS TOO LARGE. OVERFLOW.
0559 C2FA 0D                       SEC                 FLAG ERROR.
0560 C2FB 7E C2 48                 JMP    FLTADD10     RETURN.
0561                     *
0562                     *
0563                     *
```

```
                            TTL    FLTSUB
                    ********************************************************************************
                    *                                                                              *
                    *                    FLOATING POINT SUBTRACT SUBROUTINE                        *
                    *                                                                              *
                    *       This subroutine performs floating point subtraction ( FPACC1-FPACC2)   *
                    *       by inverting the sign of FPACC2 and then calling FLTADD since          *
                    *       FLTADD performs complete signed addition.  Upon returning from         *
                    *       FLTADD the sign of FPACC2 is again inverted to leave it unchanged      *
                    *       from its original value.                                               *
                    *                                                                              *
                    *                          WORSE CASE = 1062 CYCLES = 531 uS @ 2MHz            *
                    *                                                                              *
                    ********************************************************************************
                    *
                    *
C2FE                FLTSUB  EQU    *
C2FE 8D 03                  BSR    FLTSUB1      INVERT SIGN.
C300 BD C2 3C               JSR    FLTADD       GO DO FLOATING POINT ADD.
C303 96 09          FLTSUB1 LDAA   MANTSGN2     GET FPACC2 MANTISSA SIGN.
C305 88 FF                  EORA   #$FF         INVERT THE SIGN.
C307 97 09                  STAA   MANTSGN2     PUT BACK.
C309 39                     RTS                 RETURN.
                    *
                    *
                    *
```

```
0590                            TTL    FLTDIV
0591                    ********************************************************************************
0592                    *                                                                              *
0593                    *                          FLOATING POINT DIVIDE                               *
0594                    *                                                                              *
0595                    *       This subroutine performs signed floating point divide. The             *
0596                    *       operation performed is FPACC1/FPACC2.  The divisor (FPACC2) is left    *
0597                    *       unaltered and the answer is placed in FPACC1.  There are several       *
0598                    *       error conditions that can be returned by this routine.  They are:      *
0599                    *       a) division by zero.  b) overflow.  c) underflow.  As with all         *
0600                    *       other routines, an error is indicated by the carry being set and       *
0601                    *       the error code being in the A-reg.                                     *
0602                    *                                                                              *
0603                    *                    WORSE CASE = 2911 CYCLES = 1455 uS @ 2MHz                 *
0604                    *                                                                              *
0605                    ********************************************************************************
0606                    *
0607                    *
0608 C30A              FLTDIV  EQU    *
0609 C30A CE 00 05             LDX    #FPACC2EX     POINT TO FPACC2.
0610 C30D BD C1 80             JSR    CHCK0         IS THE DIVISOR 0?
0611 C310 26 04                BNE    FLTDIV1       NO. GO SEE IF THE DIVIDEND IS ZERO.
0612 C312 86 04                LDAA   #DIV0ERR      YES. RETURN A DIVIDE BY ZERO ERROR.
0613 C314 0D                   SEC                  FLAG ERROR.
0614 C315 39                   RTS                  RETURN.
0615 C316 CE 00 00      FLTDIV1 LDX    #FPACC1EX     POINT TO FPACC1.
0616 C319 BD C1 80             JSR    CHCK0         IS THE DIVIDEND 0?
0617 C31C 26 02                BNE    FLTDIV2       NO. GO PERFORM THE DIVIDE.
0618 C31E 0C                   CLC                  YES. ANSWER IS ZERO. NO ERRORS.
0619 C31F 39                   RTS                  RETURN.
0620 C320 BD C8 39      FLTDIV2 JSR    PSHFPAC2      SAVE FPACC2.
0621 C323 96 09                LDAA   MANTSGN2      GET FPACC2 MANTISSA SIGN.
0622 C325 98 04                EORA   MANTSGN1      SET THE SIGN OF THE RESULT.
0623 C327 97 04                STAA   MANTSGN1      SAVE THE RESULT.
0624 C329 CE 00 00             LDX    #0            SET UP WORK SPACE ON THE STACK.
0625 C32C 3C                   PSHX
0626 C32D 3C                   PSHX
0627 C32E 3C                   PSHX
0628 C32F 86 18                LDAA   #24           PUT LOOP COUNT ON STACK.
0629 C331 36                   PSHA
0630 C332 30                   TSX                  SET UP POINTER TO WORK SPACE.
0631 C333 DC 01                LDD    FPACC1MN      COMPARE FPACC1 & FPACC2 MANTISSAS.
0632 C335 1A 93 06             CPD    FPACC2MN      ARE THE UPPER 16 BITS THE SAME?
0633 C338 26 04                BNE    FLTDIV3       NO.
0634 C33A 96 03                LDAA   FPACC1MN+2    YES. COMPARE THE LOWER 8 BITS.
0635 C33C 91 08                CMPA   FPACC2MN+2
0636 C33E 24 10         FLTDIV3 BHS    FLTDIV4       IS FPACC2 MANTISSA > FPACC1 MANTISSA? NO.
0637 C340 7C 00 05             INC    FPACC2EX      ADD 1 TO THE EXPONENT TO KEEP NUMBER THE SAME.
0638                    *                            DID OVERFLOW OCCUR?
0639 C343 26 19                BNE    FLTDIV14      NO. GO SHIFT THE MANTISSA RIGHT 1 BIT.
0640 C345 86 02         FLTDIV8 LDAA   #OVFERR       YES. GET ERROR CODE.
0641 C347 0D                   SEC                  FLAG ERROR.
0642 C348 38            FLTDIV6 PULX                 REMOVE WORKSPACE FROM STACK.
0643 C349 38                   PULX
0644 C34A 38                   PULX
0645 C34B 31                   INS
0646 C34C BD C8 43             JSR    PULFPAC2      RESTORE FPACC2.
0647 C34F 39                   RTS                  RETURN.
0648 C350 DC 02         FLTDIV4 LDD    FPACC1MN+1    DO AN INITIAL SUBTRACT IF DIVIDEND MANTISSA IS
0649 C352 93 07                SUBD   FPACC2MN+1    GREATER THAN DIVISOR MANTISSA.
0650 C354 DD 02                STD    FPACC1MN+1
0651 C356 96 01                LDAA   FPACC1MN
0652 C358 92 06                SBCA   FPACC2MN
0653 C35A 97 01                STAA   FPACC1MN
0654 C35C 6A 00                DEC    0,X            SUBTRACT 1 FROM THE LOOP COUNT.
```

```
0655 C35E 74 00 06          FLTDIV14 LSR   FPACC2MN     SHIFT THE DIVISOR TO THE RIGHT 1 BIT.
0656 C361 76 00 07                   ROR   FPACC2MN+1
0657 C364 76 00 08                   ROR   FPACC2MN+2
0658 C367 96 00                      LDAA  FPACC1EX     GET FPACC1 EXPONENT.
0659 C369 D6 05                      LDAB  FPACC2EX     GET FPACC2 EXPONENT.
0660 C36B 50                         NEGB               ADD THE TWO'S COMPLEMENT TO SET FLAGS PROPERLY.
0661 C36C 1B                         ABA
0662 C36D 2B 06                      BMI   FLTDIV5      IF RESULT MINUS CHECK CARRY FOR POSS. OVERFLOW.
0663 C36F 25 06                      BCS   FLTDIV7      IF PLUS & CARRY SET ALL IS OK.
0664 C371 86 03                      LDAA  #UNFERR      IF NOT, UNDERFLOW ERROR.
0665 C373 20 D3                      BRA   FLTDIV6      RETURN WITH ERROR.
0666 C375 25 CE             FLTDIV5  BCS   FLTDIV8      IF MINUS & CARRY SET OVERFLOW ERROR.
0667 C377 8B 81             FLTDIV7  ADDA  #$81         ADD BACK BIAS+1 (THE '1' COMPENSATES FOR ALGOR.)
0668 C379 97 00                      STAA  FPACC1EX     SAVE RESULT.
0669 C37B DC 01             FLTDIV9  LDD   FPACC1MN     SAVE DIVIDEND IN CASE SUBTRACTION DOESN'T GO.
0670 C37D ED 04                      STD   4,X
0671 C37F 96 03                      LDAA  FPACC1MN+2
0672 C381 A7 06                      STAA  6,X
0673 C383 DC 02                      LDD   FPACC1MN+1   GET LOWER 16 BITS FOR SUBTRACTION.
0674 C385 93 07                      SUBD  FPACC2MN+1
0675 C387 DD 02                      STD   FPACC1MN+1   SAVE RESULT.
0676 C389 96 01                      LDAA  FPACC1MN     GET HIGH 8 BITS.
0677 C38B 92 06                      SBCA  FPACC2MN
0678 C38D 97 01                      STAA  FPACC1MN
0679 C38F 2A 08                      BPL   FLTDIV10     SUBTRACTION WENT OK. GO DO SHIFTS.
0680 C391 EC 04                      LDD   4,X          RESTORE OLD DIVIDEND.
0681 C393 DD 01                      STD   FPACC1MN
0682 C395 A6 06                      LDAA  6,X
0683 C397 97 03                      STAA  FPACC1MN+2
0684 C399 69 03             FLTDIV10 ROL   3,X          ROTATE CARRY INTO QUOTIENT.
0685 C39B 69 02                      ROL   2,X
0686 C39D 69 01                      ROL   1,X
0687 C39F 78 00 03                   LSL   FPACC1MN+2   SHIFT DIVIDEND TO LEFT FOR NEXT SUBTRACT.
0688 C3A2 79 00 02                   ROL   FPACC1MN+1
0689 C3A5 79 00 01                   ROL   FPACC1MN
0690 C3A8 6A 00                      DEC   0,X          DONE YET?
0691 C3AA 26 CF                      BNE   FLTDIV9      NO. KEEP GOING.
0692 C3AC 63 01                      COM   1,X          RESULT MUST BE COMPLEMENTED.
0693 C3AE 63 02                      COM   2,X
0694 C3B0 63 03                      COM   3,X
0695 C3B2 DC 02                      LDD   FPACC1MN+1   DO 1 MORE SUBTRACT FOR ROUNDING.
0696 C3B4 93 07                      SUBD  FPACC2MN+1   ( DON'T NEED TO SAVE THE RESULT. )
0697 C3B6 96 01                      LDAA  FPACC1MN
0698 C3B8 92 06                      SBCA  FPACC2MN     ( NO NEED TO SAVE THE RESULT. )
0699 C3BA EC 02                      LDD   2,X          GET LOW 16 BITS.
0700 C3BC 24 03                      BCC   FLTDIV11     IF IT DIDNT GO RESULT OK AS IS.
0701 C3BE 0C                         CLC                CLEAR THE CARRY.
0702 C3BF 20 03                      BRA   FLTDIV13     GO SAVE THE NUMBER.
0703 C3C1 C3 00 01          FLTDIV11 ADDD  #1           ROUND UP BY 1.
0704 C3C4 DD 02             FLTDIV13 STD   FPACC1MN+1   PUT IT IN FPACC1.
0705 C3C6 A6 01                      LDAA  1,X          GET HIGH 8 BITS.
0706 C3C8 89 00                      ADCA  #0
0707 C3CA 97 01                      STAA  FPACC1MN     SAVE RESULT.
0708 C3CC 24 09                      BCC   FLTDIV12     IF CARRY CLEAR ANSWER OK.
0709 C3CE 76 00 01                   ROR   FPACC1MN     IF NOT OVERFLOW. ROTATE CARRY IN.
0710 C3D1 76 00 02                   ROR   FPACC1MN+1
0711 C3D4 76 00 03                   ROR   FPACC1MN+2
0712 C3D7 0C                FLTDIV12 CLC                NO ERRORS.
0713 C3D8 7E C3 48                   JMP   FLTDIV6      RETURN.
0714                        *
0715                        *
0716                        *
```

```
                                TTL    FLTASC
                        ********************************************************************************
                        *                                                                              *
                        *                 FLOATING POINT TO ASCII CONVERSION SUBROUTINE                *
                        *                                                                              *
                        *  This subroutine performs floating point to ASCII conversion of              *
                        *  the number in FPACC1.  The ascii string is placed in a buffer               *
                        *  pointed to by the X index register.  The buffer must be at least            *
                        *  14 bytes long to contain the ASCII conversion.  The resulting               *
                        *  ASCII string is terminated by a zero (0) byte.  Upon exit the               *
                        *  X Index register will be pointing to the first character of the             *
                        *  string.  FPACC1 and FPACC2 will remain unchanged.                           *
                        *                                                                              *
                        ********************************************************************************
                        *
                        *
C3DB                    FLTASC  EQU   *
C3DB 3C                         PSHX                  SAVE THE POINTER TO THE STRING BUFFER.
C3DC CE 00 00                   LDX   #FPACC1EX       POINT TO FPACC1.
C3DF BD C1 80                   JSR   CHCKO           IS FPACC1 0?
C3E2 26 07                      BNE   FLTASC1         NO. GO CONVERT THE NUMBER.
C3E4 38                         PULX                  RESTORE POINTER.
C3E5 CC 30 00                   LDD   #$3000          GET ASCII CHARACTER + TERMINATING BYTE.
C3E8 ED 00                      STD   0,X             PUT IT IN THE BUFFER.
C3EA 39                         RTS                   RETURN.
C3EB DE 00              FLTASC1 LDX   FPACC1EX        SAVE FPACC1.
C3ED 3C                         PSHX
C3EE DE 02                      LDX   FPACC1MN+1
C3F0 3C                         PSHX
C3F1 96 04                      LDAA  MANTSGN1
C3F3 36                         PSHA
C3F4 BD C8 39                   JSR   PSHFPAC2        SAVE FPACC2.
C3F7 CE 00 00                   LDX   #0
C3FA 3C                         PSHX                  ALLOCATE LOCALS.
C3FB 3C                         PSHX
C3FC 3C                         PSHX                  SAVE SPACE FOR STRING BUFFER POINTER.
C3FD 18 30                      TSY                   POINT TO LOCALS.
C3FF CD EE 0F                   LDX   15,Y            GET POINTER FROM STACK.
C402 86 20                      LDAA  #$20            PUT A SPACE IN THE BUFFER IF NUMBER NOT NEGATIVE.
C404 7D 00 04                   TST   MANTSGN1        IS IT NEGATIVE?
C407 27 05                      BEQ   FLTASC2         NO. GO PUT SPACE.
C409 7F 00 04                   CLR   MANTSGN1        MAKE NUMBER POSITIVE FOR REST OF CONVERSION.
C40C 86 2D                      LDAA  #'-             YES. PUT MINUS SIGN IN BUFFER.
C40E A7 00              FLTASC2 STAA  0,X
C410 08                         INX                   POINT TO NEXT LOCATION.
C411 CD EF 00                   STX   0,Y             SAVE POINTER.
C414 CE C5 45           FLTASC5 LDX   #N9999999       POINT TO CONSTANT 9999999.
C417 BD C8 66                   JSR   GETFPAC2        GET INTO FPACC2.
C41A BD C5 4D                   JSR   FLTCMP          COMPARE THE NUMBERS. IS FPACC1 > 9999999?
C41D 22 19                      BHI   FLTASC3         YES. GO DIVIDE FPACC1 BY 10.
C41F CE C5 41                   LDX   #P9999999       POINT TO CONSTANT 999999.9
C422 BD C8 66                   JSR   GETFPAC2        MOVE IT INTO FPACC2.
C425 BD C5 4D                   JSR   FLTCMP          COMPARE NUMBERS. IS FPACC1 > 999999.9?
C428 22 16                      BHI   FLTASC4         YES. GO CONTINUE THE CONVERSION.
C42A 18 6A 02                   DEC   2,Y             DECREMENT THE MULT./DIV. COUNT.
C42D CE C1 8F                   LDX   #CONST10        NO. MULTIPLY BY 10. POINT TO CONSTANT.
C430 BD C8 66           FLTASC6 JSR   GETFPAC2        MOVE IT INTO FPACC2.
C433 BD C1 93                   JSR   FLTMUL
C436 20 DC                      BRA   FLTASC5         GO DO COMPARE AGAIN.
C438 18 6C 02           FLTASC3 INC   2,Y             INCREMENT THE MULT./DIV. COUNT.
C43B CE C1 8B                   LDX   #CONSTP1        POINT TO CONSTANT ".1".
C43E 20 F0                      BRA   FLTASC6         GO DIVIDE FPACC1 BY 10.
C440 CE C5 49           FLTASC4 LDX   #CONSTP5        POINT TO CONSTANT OF ".5".
C443 BD C8 66                   JSR   GETFPAC2        MOVE IT INTO FPACC2.
C446 BD C2 3C                   JSR   FLTADD          ADD .5 TO NUMBER IN FPACC1 TO ROUND IT.
```

```
0782 C449 D6 00                    LDAB   FPACC1EX     GET FPACC1 EXPONENT.
0783 C44B C0 81                    SUBB   #$81         TAKE OUT BIAS +1.
0784 C44D 50                       NEGB               MAKE IT NEGATIVE.
0785 C44E CB 17                    ADDB   #23          ADD IN THE NUMBER OF MANTISSA BITS -1.
0786 C450 20 0A                    BRA    FLTASC17     GO CHECK TO SEE IF WE NEED TO SHIFT AT ALL.
0787 C452 74 00 01          FLTASC7 LSR   FPACC1MN     SHIFT MANTISSA TO THE RIGHT BY THE RESULT (MAKE
0788 C455 76 00 02                 ROR    FPACC1MN+1   THE NUMBER AN INTEGER).
0789 C458 76 00 03                 ROR    FPACC1MN+2
0790 C45B 5A                       DECB               DONE SHIFTING?
0791 C45C 26 F4             FLTASC17 BNE  FLTASC7      NO. KEEP GOING.
0792 C45E 86 01                    LDAA   #1           GET INITIAL VALUE OF "DIGITS AFTER D.P." COUNT.
0793 C460 18 A7 03                 STAA   3,Y          INITIALIZE IT.
0794 C463 18 A6 02                 LDAA   2,Y          GET DECIMAL EXPONENT.
0795 C466 8B 08                    ADDA   #8           ADD THE NUMBER OF DECIMAL +1 TO THE EXPONENT.
0796                        *                          WAS THE ORIGINAL NUMBER > 9999999?
0797 C468 2B 0A                    BMI    FLTASC8      YES. MUST BE REPRESENTED IN SCIENTIFIC NOTATION.
0798 C46A 81 08                    CMPA   #8           WAS THE ORIGINAL NUMBER < 1?
0799 C46C 24 06                    BHS    FLTASC8      YES. MUST BE REPRESENTED IN SCIENTIFIC NOTATION.
0800 C46E 4A                       DECA               NO. NUMBER CAN BE REPRESENTED IN 7 DIGITS.
0801 C46F 18 A7 03                 STAA   3,Y          MAKE THE DECIMAL EXPONENT THE DIGIT COUNT BEFORE
0802                        *                          THE DECIMAL POINT.
0803 C472 86 02                    LDAA   #2           SETUP TO ZERO THE DECIMAL EXPONENT.
0804 C474 80 02             FLTASC8 SUBA  #2           SUBTRACT 2 FROM THE DECIMAL EXPONENT.
0805 C476 18 A7 02                 STAA   2,Y          SAVE THE DECIMAL EXPONENT.
0806 C479 18 6D 03                 TST    3,Y          DOES THE NUMBER HAVE AN INTEGER PART? (EXP. >0)
0807 C47C 2E 15                    BGT    FLTASC9      YES. GO PUT IT OUT.9
0808 C47E 86 2E                    LDAA   #'.          NO. GET DECIMAL POINT.
0809 C480 CD EE 00                 LDX    0,Y          GET POINTER TO BUFFER.
0810 C483 A7 00                    STAA   0,X          PUT THE DECIMAL POINT IN THE BUFFER.
0811 C485 08                       INX                POINT TO NEXT BUFFER LOCATION.
0812 C486 18 6D 03                 TST    3,Y          IS THE DIGIT COUNT TILL EXPONENT =0?
0813 C489 27 05                    BEQ    FLTASC18     NO. NUMBER IS <.1
0814 C48B 86 30                    LDAA   #'0          YES. FORMAT NUMBER AS .0XXXXXXX
0815 C48D A7 00                    STAA   0,X          PUT THE 0 IN THE BUFFER.
0816 C48F 08                       INX                POINT TO THE NEXT LOCATION.
0817 C490 CD EF 00          FLTASC18 STX  0,Y          SAVE NEW POINTER VALUE.
0818 C493 CE C5 2C          FLTASC9 LDX   #DECDIG      POINT TO THE TABLE OF DECIMAL DIGITS.
0819 C496 86 07                    LDAA   #7           INITIALIZE THE THE NUMBER OF DIGITS COUNT.
0820 C498 18 A7 05                 STAA   5,Y
0821 C49B 18 6F 04          FLTASC10 CLR  4,Y          CLEAR THE DECIMAL DIGIT ACCUMULATOR.
0822 C49E DC 02             FLTASC11 LDD  FPACC1MN+1   GET LOWER 16 BITS OF MANTISSA.
0823 C4A0 A3 01                    SUBD   1,X          SUBTRACT LOWER 16 BITS OF CONSTANT.
0824 C4A2 DD 02                    STD    FPACC1MN+1   SAVE RESULT.
0825 C4A4 96 01                    LDAA   FPACC1MN     GET UPPER 8 BITS.
0826 C4A6 A2 00                    SBCA   0,X          SUBTRACT UPPER 8 BITS.
0827 C4A8 97 01                    STAA   FPACC1MN     SAVE RESULT. UNDERFLOW?
0828 C4AA 25 05                    BCS    FLTASC12     YES. GO ADD DECIMAL NUMBER BACK IN.
0829 C4AC 18 6C 04                 INC    4,Y          ADD 1 TO DECIMAL NUMBER.
0830 C4AF 20 ED                    BRA    FLTASC11     TRY ANOTHER SUBTRACTION.
0831 C4B1 DC 02             FLTASC12 LDD  FPACC1MN+1   GET FPACC1 MANTISSA LOW 16 BITS.
0832 C4B3 E3 01                    ADDD   1,X          ADD LOW 16 BITS BACK IN.
0833 C4B5 DD 02                    STD    FPACC1MN+1   SAVE THE RESULT.
0834 C4B7 96 01                    LDAA   FPACC1MN     GET HIGH 8 BITS.
0835 C4B9 A9 00                    ADCA   0,X          ADD IN HIGH 8 BITS OF CONSTANT.
0836 C4BB 97 01                    STAA   FPACC1MN     SAVE RESULT.
0837 C4BD 18 A6 04                 LDAA   4,Y          GET DIGIT.
0838 C4C0 8B 30                    ADDA   #$30         MAKE IT ASCII.
0839 C4C2 3C                       PSHX               SAVE POINTER TO CONSTANTS.
0840 C4C3 CD EE 00                 LDX    0,Y          GET POINTER TO BUFFER.
0841 C4C6 A7 00                    STAA   0,X          PUT DIGIT IN BUFFER.
0842 C4C8 08                       INX                POINT TO NEXT BUFFER LOCATION.
0843 C4C9 18 6A 03                 DEC    3,Y          SHOULD WE PUT A DECIMAL POINT IN THE BUFFER YET?
0844 C4CC 26 05                    BNE    FLTASC16     NO. CONTINUE THE CONVERSION.
0845 C4CE 86 2E                    LDAA   #'.          YES. GET DECIMAL POINT.
0846 C4D0 A7 00                    STAA   0,X          PUT IT IN THE BUFFER.
0847 C4D2 08                       INX                POINT TO THE NEXT BUFFER LOCATION.
```

```
0848 C4D3 CD EF 00          FLTASC16 STX    0,Y            SAVE UPDATED POINTER.
0849 C4D6 38                         PULX                  RESTORE POINTER TO CONSTANTS.
0850 C4D7 08                         INX                   POINT TO NEXT CONSTANT.
0851 C4D8 08                         INX
0852 C4D9 08                         INX
0853 C4DA 18 6A 05                   DEC    5,Y            DONE YET?
0854 C4DD 26 BC                      BNE    FLTASC10       NO. CONTINUE CONVERSION OF "MANTISSA".
0855 C4DF CD EE 00                   LDX    0,Y            YES. POINT TO BUFFER STRING BUFFER.
0856 C4E2 09                FLTASC13 DEX                    POINT TO LAST CHARACTER PUT IN THE BUFFER.
0857 C4E3 A6 00                      LDAA   0,X            GET IT.
0858 C4E5 81 30                      CMPA   #$30           WAS IT AN ASCII 0?
0859 C4E7 27 F9                      BEQ    FLTASC13       YES. REMOVE TRAILING ZEROS.
0860 C4E9 08                         INX                   POINT TO NEXT AVAILABLE LOCATION IN BUFFER.
0861 C4EA 18 E6 02                   LDAB   2,Y            DO WE NEED TO PUT OUT AN EXPONENT?
0862 C4ED 27 2A                      BEQ    FLTASC15       NO. WE'RE DONE.
0863 C4EF 86 45                      LDAA   #'E            YES. PUT AN 'E' IN THE BUFFER.
0864 C4F1 A7 00                      STAA   0,X
0865 C4F3 08                         INX                   POINT TO NEXT BUFFER LOCATION.
0866 C4F4 86 2B                      LDAA   #'+            ASSUME EXPONENT IS POSITIVE.
0867 C4F6 A7 00                      STAA   0,X            PUT PLUS SIGN IN THE BUFFER.
0868 C4F8 5D                         TSTB                  IS IT REALLY MINUS?
0869 C4F9 2A 05                      BPL    FLTASC14       NO. IS'S OK AS IS.
0870 C4FB 50                         NEGB                  YES. MAKE IT POSITIVE.
0871 C4FC 86 2D                      LDAA   #'-           PUT THE MINUS SIGN IN THE BUFFER.
0872 C4FE A7 00                      STAA   0,X
0873 C500 08                FLTASC14 INX                    POINT TO NEXT BUFFER LOCATION.
0874 C501 CD EF 00                   STX    0,Y            SAVE POINTER TO STRING BUFFER.
0875 C504 4F                         CLRA                  SET UP FOR DIVIDE.
0876 C505 CE 00 0A                   LDX    #10            DIVIDE DECIMAL EXPONENT BY 10.
0877 C508 02                         IDIV
0878 C509 37                         PSHB                  SAVE REMAINDER.
0879 C50A 8F                         XGDX                  PUT QUOTIENT IN D.
0880 C50B CB 30                      ADDB   #$30           MAKE IT ASCII.
0881 C50D CD EE 00                   LDX    0,Y            GET POINTER.
0882 C510 E7 00                      STAB   0,X            PUT NUMBER IN BUFFER.
0883 C512 08                         INX                   POINT TO NEXT LOCATION.
0884 C513 33                         PULB                  GET SECOND DIGIT.
0885 C514 CB 30                      ADDB   #$30           MAKE IT ASCII.
0886 C516 E7 00                      STAB   0,X            PUT IT IN THE BUFFER.
0887 C518 08                         INX                   POINT TO NEXT LOCATION.
0888 C519 6F 00             FLTASC15 CLR    0,X            TERMINATE STRING WITH A ZERO BYTE.
0889 C51B 38                         PULX                  CLEAR LOCALS FROM STACK.
0890 C51C 38                         PULX
0891 C51D 38                         PULX
0892 C51E BD C8 43                   JSR    PULFPAC2       RESTORE FPACC2.
0893 C521 32                         PULA
0894 C522 97 04                      STAA   MANTSGN1
0895 C524 38                         PULX                  RESTORE FPACC1.
0896 C525 DF 02                      STX    FPACC1MN+1
0897 C527 38                         PULX
0898 C528 DF 00                      STX    FPACC1EX
0899 C52A 38                         PULX                  POINT TO THE START OF THE ASCII STRING.
0900 C52B 39                         RTS                   RETURN.
0901                        *
0902                        *
0903 C52C                   DECDIG   EQU    *
0904 C52C 0F 42 40                   FCB    $0F,$42,$40    DECIMAL 1,000,000
0905 C52F 01 86 A0                   FCB    $01,$86,$A0    DECIMAL   100,000
0906 C532 00 27 10                   FCB    $00,$27,$10    DECIMAL    10,000
0907 C535 00 03 E8                   FCB    $00,$03,$E8    DECIMAL     1,000
0908 C538 00 00 64                   FCB    $00,$00,$64    DECIMAL       100
0909 C53B 00 00 0A                   FCB    $00,$00,$0A    DECIMAL        10
0910 C53E 00 00 01                   FCB    $00,$00,$01    DECIMAL         1
0911                        *
0912                        *
0913 C541                   P9999999 EQU    *              CONSTANT 999999.9
```

```
0914 C541 94 74 23 FE            FCB   $94,$74,$23,$FE
0915                             *
0916 C545                        N9999999 EQU   *          CONSTANT 9999999.
0917 C545 98 18 96 7F            FCB   $98,$18,$96,$7F
0918                             *
0919 C549                        CONSTP5 EQU   *          CONSTANT .5
0920 C549 80 00 00 00            FCB   $80,$00,$00,$00
0921                             *
0922                             *
0923 C54D                        FLTCMP   EQU   *
0924 C54D 7D 00 04               TST   MANTSGN1     IS FPACC1 NEGATIVE?
0925 C550 2A 12                  BPL   FLTCMP2      NO. CONTINUE WITH COMPARE.
0926 C552 7D 00 09               TST   MANTSGN2     IS FPACC2 NEGATIVE?
0927 C555 2A 0D                  BPL   FLTCMP2      NO. CONTINUE WITH COMPARE.
0928 C557 DC 05                  LDD   FPACC2EX     YES. BOTH ARE NEGATIVE SO COMPARE MUST BE DONE
0929 C559 1A 93 00               CPD   FPACC1EX     BACKWARDS. ARE THEY EQUAL SO FAR?
0930 C55C 26 05                  BNE   FLTCMP1      NO. RETURN WITH CONDITION CODES SET.
0931 C55E DC 07                  LDD   FPACC2MN+1   YES. COMPARE LOWER 16 BITS OF MANTISSAS.
0932 C560 1A 93 02               CPD   FPACC1MN+1
0933 C563 39                     FLTCMP1  RTS                RETURN WITH CONDITION CODES SET.
0934 C564 96 04                  FLTCMP2  LDAA  MANTSGN1     GET FPACC1 MANTISSA SIGN.
0935 C566 91 09                  CMPA  MANTSGN2     BOTH POSITIVE?
0936 C568 26 F9                  BNE   FLTCMP1      NO. RETURN WITH CONDITION CODES SET.
0937 C56A DC 00                  LDD   FPACC1EX     GET FPACC1 EXPONENT & UPPER 8 BITS OF MANTISSA.
0938 C56C 1A 93 05               CPD   FPACC2EX     SAME AS FPACC2?
0939 C56F 26 F2                  BNE   FLTCMP1      NO. RETURN WITH CONDITION CODES SET.
0940 C571 DC 02                  LDD   FPACC1MN+1   GET FPACC1 LOWER 16 BITS OF MANTISSA.
0941 C573 1A 93 07               CPD   FPACC2MN+1   COMPARE WITH FPACC2 LOWER 16 BITS OF MANTISSA.
0942 C576 39                     RTS                RETURN WITH CONDITION CODES SET.
0943                             *
0944                             *
0945                             *
```

```
                                TTL    INT2FLT
                        ********************************************************************************
                        *                                                                              *
                        *                        UNSIGNED INTEGER TO FLOATING POINT                    *
                        *                                                                              *
                        *       This subroutine performs "unsigned" integer to floating point          *
                        *       conversion of a 16 bit word.  The 16 bit integer must be in the        *
                        *       lower 16 bits of FPACC1 mantissa.  The resulting floating point        *
                        *       number is returned in FPACC1.                                          *
                        *                                                                              *
                        ********************************************************************************
                        *
                        *
C577                    UINT2FLT EQU    *
C577 CE 00 00                    LDX    #FPACC1EX      POINT TO FPACC1.
C57A BD C1 80                    JSR    CHCKO          IS IT ALREADY 0?
C57D 26 01                       BNE    UINTFLT1       NO. GO CONVERT.
C57F 39                          RTS                   YES. JUST RETURN.
C580 86 98              UINTFLT1 LDAA   #$98            GET BIAS + NUMBER OF BITS IN MANTISSA.
C582 97 00                       STAA   FPACC1EX       INITIALIZE THE EXPONENT.
C584 BD C1 61                    JSR    FPNORM         GO MAKE IT A NORMALIZED FLOATING POINT VALUE.
C587 0C                          CLC                   NO ERRORS.
C588 39                          RTS                   RETURN.
                        *
                        *
                        *
                        ********************************************************************************
                        *                                                                              *
                        *                        SIGNED INTEGER TO FLOATING POINT                      *
                        *                                                                              *
                        *       This routine works just like the unsigned integer to floating          *
                        *       point routine except the the 16 bit integer in the FPACC1              *
                        *       mantissa is considered to be in two's complement format.  This         *
                        *       will return a floating point number in the range -32768 to +32767.     *
                        *                                                                              *
                        ********************************************************************************
                        *
                        *
C589                    SINT2FLT EQU    *
C589 DC 02                       LDD    FPACC1MN+1     GET THE LOWER 16 BITS OF FPACC1 MANTISSA.
C58B 36                          PSHA                  SAVE SIGN OF NUMBER.
C58C 2A 07                       BPL    SINTFLT1       IF POSITIVE JUST GO CONVERT.
C58E 43                          COMA                  MAKE POSITIVE.
C58F 53                          COMB
C590 C3 00 01                    ADDD   #1             TWO'S COMPLEMENT.
C593 DD 02                       STD    FPACC1MN+1     PUT IT BACK IN FPACC1 MANTISSA.
C595 8D E0              SINTFLT1 BSR    UINT2FLT       GO CONVERT.
C597 32                          PULA                  GET SIGN OF ORIGINAL INTEGER.
C598 C6 FF                       LDAB   #$FF           GET "MINUS SIGN".
C59A 4D                          TSTA                  WAS THE NUMBER NEGATIVE?
C59B 2A 02                       BPL    SINTFLT2       NO. RETURN.
C59D D7 04                       STAB   MANTSGN1       YES. SET FPACC1 SIGN BYTE.
C59F 0C                 SINTFLT2 CLC                   NO ERRORS.
C5A0 39                          RTS                   RETURN.
                        *
                        *
                        *
```

```
1003                                    TTL    FLT2INT
1004                    ********************************************************************************
1005                    *                                                                              *
1006                    *                    FLOATING POINT TO INTEGER CONVERSION                      *
1007                    *                                                                              *
1008                    *       This subroutine will perform "unsigned" floating point to integer      *
1009                    *       conversion.  The floating point number if positive, will be            *
1010                    *       converted to an unsigned 16 bit integer ( 0 <= X <= 65535 ).  If       *
1011                    *       the number is negative it will be converted to a twos complement       *
1012                    *       16 bit integer.  This type of conversion will allow 16 bit             *
1013                    *       addresses to be represented as positive numbers when in floating       *
1014                    *       point format.  Any fractional number part is disguarded                *
1015                    *                                                                              *
1016                    ********************************************************************************
1017                    *
1018                    *
1019 C5A1              FLT2INT  EQU    *
1020 C5A1 CE 00 00              LDX    #FPACC1EX     POINT TO FPACC1.
1021 C5A4 BD C1 80              JSR    CHCK0         IS IT 0?
1022 C5A7 27 41                 BEQ    FLT2INT3      YES. JUST RETURN.
1023 C5A9 D6 00                 LDAB   FPACC1EX      GET FPACC1 EXPONENT.
1024 C5AB C1 81                 CMPB   #$81          IS THERE AN INTEGER PART?
1025 C5AD 25 34                 BLO    FLT2INT2      NO. GO PUT A 0 IN FPACC1.
1026 C5AF 7D 00 04              TST    MANTSGN1      IS THE NUMBER NEGATIVE?
1027 C5B2 2B 16                 BMI    FLT2INT1      YES. GO CONVERT NEGATIVE NUMBER.
1028 C5B4 C1 90                 CMPB   #$90          IS THE NUMBER TOO LARGE TO BE MADE AN INTEGER?
1029 C5B6 22 27                 BHI    FLT2INT4      YES. RETURN WITH AN ERROR.
1030 C5B8 C0 98                 SUBB   #$98          SUBTRACT THE BIAS PLUS THE NUMBER OF BITS.
1031 C5BA 74 00 01       FLT2INT5 LSR    FPACC1MN      MAKE THE NUMBER AN INTEGER.
1032 C5BD 76 00 02              ROR    FPACC1MN+1
1033 C5C0 76 00 03              ROR    FPACC1MN+2
1034 C5C3 5C                    INCB                 DONE SHIFTING?
1035 C5C4 26 F4                 BNE    FLT2INT5      NO. KEEP GOING.
1036 C5C6 7F 00 00              CLR    FPACC1EX      ZERO THE EXPONENT (ALSO CLEARS THE CARRY).
1037 C5C9 39                    RTS
1038 C5CA C1 8F          FLT2INT1 CMPB   #$8F          IS THE NUMBER TOO SMALL TO BE MADE AN INTEGER?
1039 C5CC 22 11                 BHI    FLT2INT4      YES. RETURN ERROR.
1040 C5CE C0 98                 SUBB   #$98          SUBTRACT BIAS PLUS NUMBER OF BITS.
1041 C5D0 8D E8                 BSR    FLT2INT5      GO DO SHIFT.
1042 C5D2 DC 02                 LDD    FPACC1MN+1    GET RESULTING INTEGER.
1043 C5D4 43                    COMA                 MAKE IT NEGATIVE.
1044 C5D5 53                    COMB
1045 C5D6 C3 00 01              ADDD   #1            TWO'S COMPLEMENT.
1046 C5D9 DD 02                 STD    FPACC1MN+1    SAVE RESULT.
1047 C5DB 7F 00 04              CLR    MANTSGN1      CLEAR MANTISSA SIGN. (ALSO CLEARS THE CARRY)
1048 C5DE 39                    RTS                  RETURN.
1049 C5DF 86 05          FLT2INT4 LDAA   #TOLGSMER     NUMBER TOO LARGE OR TOO SMALL TO CONVERT TO INT.
1050 C5E1 0D                    SEC                  FLAG ERROR.
1051 C5E2 39                    RTS                  RETURN.
1052 C5E3 CC 00 00       FLT2INT2 LDD    #0
1053 C5E6 DD 00                 STD    FPACC1EX      ZERO FPACC1.
1054 C5E8 DD 02                 STD    FPACC1MN+1    (ALSO CLEARS THE CARRY)
1055 C5EA 39             FLT2INT3 RTS                  RETURN.
1056                    *
1057                    *
1058                    *
```

```
                            TTL    FLTSQR
                    ********************************************************************************
                    *                                                                              *
                    *                          SQUARE ROOT SUBROUTINE                              *
                    *                                                                              *
                    *       This routine is used to calculate the square root of the floating      *
                    *       point number in FPACC1.  If the number in FPACC1 is negative an        *
                    *       error is returned.                                                     *
                    *                                                                              *
                    *                          WORSE CASE = 16354 CYCLES = 8177 uS @ 2MHz          *
                    *                                                                              *
                    ********************************************************************************
                    *
                    *
C5EB                FLTSQR  EQU    *
C5EB CE 00 00               LDX    #FPACC1EX    POINT TO FPACC1.
C5EE BD C1 80               JSR    CHCK0        IS IT ZERO?
C5F1 26 01                  BNE    FLTSQR1      NO. CHECK FOR NEGATIVE.
C5F3 39                     RTS                 YES. RETURN.
C5F4 7D 00 04       FLTSQR1 TST    MANTSGN1     IS THE NUMBER NEGATIVE?
C5F7 2A 04                  BPL    FLTSQR2      NO. GO TAKE ITS SQUARE ROOT.
C5F9 86 06                  LDAA   #NSQRTERR    YES. ERROR.
C5FB 0D                     SEC                 FLAG ERROR.
C5FC 39                     RTS                 RETURN.
C5FD BD C8 39       FLTSQR2 JSR    PSHFPAC2     SAVE FPACC2.
C600 86 04                  LDAA   #4           GET ITERATION LOOP COUNT.
C602 36                     PSHA                SAVE IT ON THE STACK.
C603 DE 02                  LDX    FPACC1MN+1   SAVE INITIAL NUMBER.
C605 3C                     PSHX
C606 DE 00                  LDX    FPACC1EX
C608 3C                     PSHX
C609 18 30                  TSY                 POINT TO IT.
C60B 8D 39                  BSR    TFR1TO2      TRANSFER FPACC1 TO FPACC2.
C60D 96 05                  LDAA   FPACC2EX     GET FPACC1 EXPONENT.
C60F 80 80                  SUBA   #$80         REMOVE BIAS FROM EXPONENT.
C611 4C                     INCA                COMPENSATE FOR ODD EXPONENTS (GIVES CLOSER GUESS)
C612 2A 03                  BPL    FLTSQR3      IF NUMBER >1 DIVIDE EXPONENT BY 2 & ADD BIAS.
C614 44                     LSRA                IF <1 JUST DIVIDE IT BY 2.
C615 20 03                  BRA    FLTSQR4      GO CALCULATE THE SQUARE ROOT.
C617 44             FLTSQR3 LSRA                DIVIDE EXPONENT BY 2.
C618 8B 80                  ADDA   #$80         ADD BIAS BACK IN.
C61A 97 05          FLTSQR4 STAA   FPACC2EX     SAVE EXPONENT/2.
C61C BD C3 0A       FLTSQR5 JSR    FLTDIV       DIVIDE THE ORIGINAL NUMBER BY THE GUESS.
C61F BD C2 3C               JSR    FLTADD       ADD THE "GUESS" TO THE QUOTIENT.
C622 7A 00 00               DEC    FPACC1EX     DIVIDE THE RESULT BY 2 TO PRODUCE A NEW GUESS.
C625 8D 1F                  BSR    TFR1TO2      PUT THE NEW GUESS INTO FPACC2.
C627 18 EC 00               LDD    0,Y          GET THE ORIGINAL NUMBER.
C62A DD 00                  STD    FPACC1EX     PUT IT BACK IN FPACC1.
C62C 18 EC 02               LDD    2,Y          GET MANTISSA LOWER 16 BITS.
C62F DD 02                  STD    FPACC1MN+1
C631 18 6A 04               DEC    4,Y          BEEN THROUGH THE LOOP 4 TIMES?
C634 26 E6                  BNE    FLTSQR5      NO. KEEP GOING.
C636 DC 05                  LDD    FPACC2EX     THE FINAL GUESS IS THE ANSWER.
C638 DD 00                  STD    FPACC1EX     PUT IT IN FPACC1.
C63A DC 07                  LDD    FPACC2MN+1
C63C DD 02                  STD    FPACC1MN+1
C63E 38                     PULX                GET RID OF ORIGINAL NUMBER.
C63F 38                     PULX
C640 31                     INS                 GET RID OF LOOP COUNT VARIABLE.
C641 BD C8 43               JSR    PULFPAC2     RESTORE FPACC2.
C644 0C                     CLC                 NO ERRORS.
C645 39                     RTS
                    *
                    *
C646                TFR1TO2 EQU    *
```

```
1124 C646 DC 00                    LDD    FPACC1EX     GET FPACC1 EXPONENT & HIGH 8 BIT OF MANTISSA.
1125 C648 DD 05                    STD    FPACC2EX     PUT IT IN FPACC2.
1126 C64A DC 02                    LDD    FPACC1MN+1   GET FPACC1 LOW 16 BITS OF MANTISSA.
1127 C64C DD 07                    STD    FPACC2MN+1   PUT IT IN FPACC2.
1128 C64E 96 04                    LDAA   MANTSGN1     TRANSFER THE SIGN.
1129 C650 97 09                    STAA   MANTSGN2
1130 C652 39                       RTS                 RETURN.
1131                         *
1132                         *
1133                         *
```

```
                            TTL    FLTSIN
*******************************************************************************
*                                                                             *
*                          FLOATING POINT SINE                                *
*                                                                             *
*******************************************************************************
*
*
C653                FLTSIN  EQU    *
C653 BD C8 39               JSR    PSHFPAC2     SAVE FPACC2 ON THE STACK.
C656 BD C7 59               JSR    ANGRED       GO REDUCE THE ANGLE TO BETWEEN +/-PI.
C659 37                     PSHB                SAVE THE QUAD COUNT.
C65A 36                     PSHA                SAVE THE SINE/COSINE FLAG.
C65B BD C8 13               JSR    DEG2RAD      CONVERT DEGREES TO RADIANS.
C65E 32                     PULA                RESTORE THE SINE/COSINE FLAG.
C65F BD C6 8F       FLTSIN1 JSR    SINCOS       GO GET THE SINE OF THE ANGLE.
C662 32                     PULA                RESTORE THE QUAD COUNT.
C663 81 02                  CMPA   #2           WAS THE ANGLE IN QUADS 1 OR 2?
C665 23 03                  BLS    FLTSIN2      YES. SIGN OF THE ANSWER IS OK.
C667 73 00 04               COM    MANTSGN1     NO. SINE IN QUADS 3 & 4 IS NEGATIVE.
C66A 0C             FLTSIN2 CLC                 SHOW NO ERRORS.
C66B BD C8 43               JSR    PULFPAC2     RESTORE FPACC2
C66E 39                     RTS                 RETURN.
*
*
*
```

```
1160                            TTL    FLTCOS
1161                    ********************************************************************************
1162                    *                                                                              *
1163                    *                          FLOATING POINT COSINE                               *
1164                    *                                                                              *
1165                    ********************************************************************************
1166                    *
1167                    *
1168 C66F               FLTCOS   EQU    *
1169 C66F BD C8 39               JSR    PSHFPAC2     SAVE FPACC2 ON THE STACK.
1170 C672 BD C7 59               JSR    ANGRED       GO REDUCE THE ANGLE TO BETWEEN +/-PI.
1171 C675 37                    PSHB                SAVE THE QUAD COUNT.
1172 C676 36                    PSHA                SAVE THE SINE/COSINE FLAG.
1173 C677 BD C8 13              JSR    DEG2RAD      CONVERT TO RADIANS.
1174 C67A 32                    PULA                RESTORE THE SINE/COSINE FLAG.
1175 C67B 88 01                 EORA   #$01         COMPLIMENT 90'S COPMLIMENT FLAG FOR COSINE.
1176 C67D BD C6 8F              JSR    SINCOS       GO GET THE COSINE OF THE ANGLE.
1177 C680 32                    PULA                RESTORE THE QUAD COUNT.
1178 C681 81 01                 CMPA   #1           WAS THE ORIGINAL ANGLE IN QUAD 1?
1179 C683 27 07                 BEQ    FLTCOS1      YES. SIGN IS OK.
1180 C685 81 04                 CMPA   #4           WAS IT IN QUAD 4?
1181 C687 27 03                 BEQ    FLTCOS1      YES. SIGN IS OK.
1182 C689 73 00 04              COM    MANTSGN1     NO. COSINE IS NEGATIVE IN QUADS 2 & 3.
1183 C68C 7E C6 6A         FLTCOS1  JMP    FLTSIN2      FLAG NO ERRORS, RESTORE FPACC2, & RETURN.
1184                    *
1185                    *
1186                    *
```

```
                                TTL    SINCOS
*******************************************************************************
*                                                                             *
*                    FLOATING POINT SINE AND COSINE SUBROUTINE                *
*                                                                             *
*******************************************************************************
*
*
C68F                    SINCOS  EQU    *
C68F 36                         PSHA                SAVE SINE/COSINE FLAG ON STACK.
C690 DE 02                      LDX    FPACC1MN+1   SAVE THE VALUE OF THE ANGLE.
C692 3C                         PSHX
C693 DE 00                      LDX    FPACC1EX
C695 3C                         PSHX
C696 96 04                      LDAA   MANTSGN1
C698 36                         PSHA
C699 CE C7 C3                   LDX    #SINFACT     POINT TO THE FACTORIAL TABLE.
C69C 3C                         PSHX                SAVE POINTER TO THE SINE FACTORIAL TABLE.
C69D 3C                         PSHX                JUST ALLOCATE ANOTHER LOCAL (VALUE NOT IMPORTANT)
C69E 86 04                      LDAA   #$4          GET INITIAL LOOP COUNT.
C6A0 36                         PSHA                SAVE AS LOCAL ON STACK
C6A1 18 30                      TSY                 POINT TO LOCALS.
C6A3 BD C6 46                   JSR    TFR1TO2      TRANSFER FPACC1 TO FPACC2.
C6A6 BD C1 93                   JSR    FLTMUL       GET X^2 IN FPACC1.
C6A9 18 6D 0A                   TST    10,Y         ARE WE DOING THE SINE?
C6AC 27 0B                      BEQ    SINCOS7      YES. GO DO IT.
C6AE CE C7 D3                   LDX    #COSFACT     NO. GET POINTER TO COSINE FACTORIAL TABLE.
C6B1 CD EF 01                   STX    1,Y          SAVE IT.
C6B4 BD C6 46                   JSR    TFR1TO2      COPY X^2 INTO FPACC2.
C6B7 20 06                      BRA    SINCOS4      GENERATE EVEN POWERS OF "X" FOR COSINE.
C6B9 BD C7 AA           SINCOS7 JSR    EXG1AND2     PUT X^2 IN FPACC2 & X IN FPACC1.
C6BC BD C1 93           SINCOS1 JSR    FLTMUL       CREATE X^3,5,7,9 OR X^2,4,6,8.
C6BF DE 02              SINCOS4 LDX    FPACC1MN+1   SAVE EACH ONE ON THE STACK.
C6C1 3C                         PSHX
C6C2 DE 00                      LDX    FPACC1EX
C6C4 3C                         PSHX
C6C5 96 04                      LDAA   MANTSGN1
C6C7 36                         PSHA                SAVE THE MANTISSA SIGN.
C6C8 18 6A 00                   DEC    0,Y          HAVE WE GENERATED ALL THE POWERS YET?
C6CB 26 EF                      BNE    SINCOS1      NO. GO DO SOME MORE.
C6CD 86 04                      LDAA   #$4          SET UP LOOP COUNT.
C6CF 18 A7 00                   STAA   0,Y
C6D2 30                         TSX                 POINT TO POWERS ON THE STACK.
C6D3 CD EF 03           SINCOS2 STX    3,Y          SAVE THE POINTER.
C6D6 CD EE 01                   LDX    1,Y          GET THE POINTER TO THE FACTORIAL CONSTANTS.
C6D9 BD C8 66                   JSR    GETFPAC2     PUT THE NUMBER IN FPACC2.
C6DC 08                         INX                 POINT TO THE NEXT CONSTANT.
C6DD 08                         INX
C6DE 08                         INX
C6DF 08                         INX
C6E0 CD EF 01                   STX    1,Y          SAVE THE POINTER.
C6E3 CD EE 03                   LDX    3,Y          GET POINTER TO POWERS.
C6E6 A6 00                      LDAA   0,X          GET NUMBER SIGN.
C6E8 97 04                      STAA   MANTSGN1     PUT IN FPACC1 MANTISSA SIGN.
C6EA EC 01                      LDD    1,X          GET LOWER 16-BITS OF THE MANTISSA.
C6EC DD 00                      STD    FPACC1EX     PUT IN FPACC1 MANTISSA.
C6EE EC 03                      LDD    3,X          GET HIGH 8 BITS OF THE MANTISSA & EXPONENT.
C6F0 DD 02                      STD    FPACC1MN+1   PUT IT IN FPACC1 EXPONENT & MANTISSA.
C6F2 BD C1 93                   JSR    FLTMUL       MULTIPLY THE TWO.
C6F5 CD EE 03                   LDX    3,Y          GET POINTER TO POWERS BACK.
C6F8 DC 02                      LDD    FPACC1MN+1   SAVE RESULT WHERE THE POWER OF X WAS.
C6FA ED 03                      STD    3,X
C6FC DC 00                      LDD    FPACC1EX
C6FE ED 01                      STD    1,X
C700 96 04                      LDAA   MANTSGN1     SAVE SIGN.
```

```
1252 C702 A7 00                STAA   0,X
1253 C704 08                   INX                  POINT TO THE NEXT POWER.
1254 C705 08                   INX
1255 C706 08                   INX
1256 C707 08                   INX
1257 C708 08                   INX
1258 C709 18 6A 00             DEC    0,Y           DONE?
1259 C70C 26 C5                BNE    SINCOS2       NO. GO DO ANOTHER MULTIPLICATION.
1260 C70E 86 03                LDAA   #$3           GET LOOP COUNT.
1261 C710 18 A7 00             STAA   0,Y           SAVE IT.
1262 C713 CD EE 03    SINCOS3  LDX    3,Y           POINT TO RESULTS ON THE STACK.
1263 C716 09                   DEX                  POINT TO PREVIOUS RESULT.
1264 C717 09                   DEX
1265 C718 09                   DEX
1266 C719 09                   DEX
1267 C71A 09                   DEX
1268 C71B CD EF 03             STX    3,Y           SAVE THE NEW POINTER.
1269 C71E A6 00                LDAA   0,X           GET NUMBERS SIGN.
1270 C720 97 09                STAA   MANTSGN2      PUT IT IN FPACC2.
1271 C722 EC 01                LDD    1,X           GET LOW 16 BITS OF THE MANTISSA.
1272 C724 DD 05                STD    FPACC2EX      PUT IN FPACC2.
1273 C726 EC 03                LDD    3,X           GET HIGH 8 BIT & EXPONENT.
1274 C728 DD 07                STD    FPACC2MN+1    PUT IN FPACC2.
1275 C72A BD C2 3C             JSR    FLTADD        GO ADD THE TWO NUMBERS.
1276 C72D 18 6A 00             DEC    0,Y           DONE?
1277 C730 26 E1                BNE    SINCOS3       NO. GO ADD THE NEXT TERM IN.
1278 C732 18 6D 0A             TST    10,Y          ARE WE DOING THE SINE?
1279 C735 27 08                BEQ    SINCOS5       YES. GO PUT THE ORIGINAL ANGLE INTO FPACC2.
1280 C737 CE C7 E3             LDX    #ONE          NO. FOR COSINE PUT THE CONSTANT 1 INTO FPACC2.
1281 C73A BD C8 66             JSR    GETFPAC2
1282 C73D 20 0F                BRA    SINCOS6       GO ADD IT TO THE SUM OF THE TERMS.
1283 C73F 18 A6 05    SINCOS5  LDAA   5,Y           GET THE VALUE OF THE ORIGINAL ANGLE.
1284 C742 97 09                STAA   MANTSGN2      PUT IT IN FPACC2.
1285 C744 18 EC 06             LDD    6,Y
1286 C747 DD 05                STD    FPACC2EX
1287 C749 18 EC 08             LDD    8,Y
1288 C74C DD 07                STD    FPACC2MN+1
1289 C74E BD C2 3C    SINCOS6  JSR    FLTADD        GO ADD IT TO THE SUM OF THE TERMS.
1290 C751 30                   TSX                  NOW CLEAN UP THE STACK.
1291 C752 8F                   XGDX                 PUT STACK IN D.
1292 C753 C3 00 1F             ADDD   #31           CLEAR ALL THE TERMS & TEMPS OFF THE STACK.
1293 C756 8F                   XGDX
1294 C757 35                   TXS                  UPDATE THE STACK POINTER.
1295 C758 39                   RTS                  RETURN.
1296                  *
1297                  *
1298 C759             ANGRED   EQU    *
1299 C759 4F                   CLRA                 INITIALIZE THE 45'S COMPLIMENT FLAG.
1300 C75A 36                   PSHA                 PUT IT ON THE STACK.
1301 C75B 4C                   INCA                 INITIALIZE THE QUAD COUNT TO 1.
1302 C75C 36                   PSHA                 PUT IT ON THE STACK.
1303 C75D 18 30                TSY                  POINT TO IT.
1304 C75F CE C7 EB             LDX    #THREE60      POINT TO THE CONSTANT 360.
1305 C762 BD C8 66             JSR    GETFPAC2      GET IT INTO FPACC.
1306 C765 7D 00 04             TST    MANTSGN1      IS THE INPUT ANGLE NEGATIVE:
1307 C768 2A 03                BPL    ANGRED1       NO. SKIP THE ADD.
1308 C76A BD C2 3C             JSR    FLTADD        YES. MAKE THE ANGLE POSITIVE BY ADDING 360 DEG.
1309 C76D 7A 00 05    ANGRED1  DEC    FPACC2EX      MAKE THE CONSTANT IN FPACC2 90 DEGREES.
1310 C770 7A 00 05             DEC    FPACC2EX
1311 C773 BD C5 4D    ANGRED2  JSR    FLTCMP        IS THE ANGLE LESS THAN 90 DEGREES ALREADY?
1312 C776 23 08                BLS    ANGRED3       YES. RETURN WITH QUAD COUNT.
1313 C778 BD C2 FE             JSR    FLTSUB        NO. REDUCE ANGLE BY 90 DEGREES.
1314 C77B 18 6C 00             INC    0,Y           INCREMENT THE QUAD COUNT.
1315 C77E 20 F3                BRA    ANGRED2       GO SEE IF IT'S LESS THAN 90 NOW.
1316 C780 18 A6 00    ANGRED3  LDAA   0,Y           GET THE QUAD COUNT.
1317 C783 81 01                CMPA   #1            WAS THE ORIGINAL ANGLE IN QUAD 1?
```

```
1318 C785 27 0B                    BEQ    ANGRED4      YES. COMPUTE TRIG FUNCTION AS IS.
1319 C787 81 03                    CMPA   #3           NO. WAS THE ORIGINAL ANGLE IN QUAD 3?
1320 C789 27 07                    BEQ    ANGRED4      YES. COMPUTE THE TRIG FUNCTION AS IF IN QUAD 1.
1321 C78B 86 FF                    LDAA   #$FF         NO. MUST COMPUTE THE TRIG FUNCTION OF THE 90'S
1322 C78D 97 04                    STAA   MANTSGN1     COMPLIMENT ANGLE.
1323 C78F BD C2 3C                 JSR    FLTADD       ADD 90 DEGREES TO THE NEGATED ANGLE.
1324 C792 7A 00 05          ANGRED4 DEC    FPACC2EX     MAKE THE ANGLE IN FPACC2 45 DEGREES.
1325 C795 BD C5 4D                 JSR    FLTCMP       IS THE ANGLE < 45 DEGREES?
1326 C798 23 0D                    BLS    ANGRED5      YES. IT'S OK AS IT IS.
1327 C79A 7C 00 05                 INC    FPACC2EX     NO. MUST GET THE 90'S COMPLIMENT.
1328 C79D 86 FF                    LDAA   #$FF         MAKE FPACC1 NEGATIVE.
1329 C79F 97 04                    STAA   MANTSGN1
1330 C7A1 BD C2 3C                 JSR    FLTADD       GET THE 90'S COMPLIMENT.
1331 C7A4 18 6C 01                 INC    1,Y          SET THE FLAG.
1332 C7A7 33                ANGRED5 PULB               GET THE QUAD COUNT.
1333 C7A8 32                       PULA               GET THE COMPLIMENT FLAG.
1334 C7A9 39                       RTS                RETURN WITH THE QUAD COUNT & COMPLIMENT FLAG.
1335                        *
1336                        *
1337 C7AA                   EXG1AND2 EQU   *
1338 C7AA DC 00                    LDD    FPACC1EX
1339 C7AC DE 05                    LDX    FPACC2EX
1340 C7AE DD 05                    STD    FPACC2EX
1341 C7B0 DF 00                    STX    FPACC1EX
1342 C7B2 DC 02                    LDD    FPACC1MN+1
1343 C7B4 DE 07                    LDX    FPACC2MN+1
1344 C7B6 DD 07                    STD    FPACC2MN+1
1345 C7B8 DF 02                    STX    FPACC1MN+1
1346 C7BA 96 04                    LDAA   MANTSGN1
1347 C7BC D6 09                    LDAB   MANTSGN2
1348 C7BE 97 09                    STAA   MANTSGN2
1349 C7C0 D7 04                    STAB   MANTSGN1
1350 C7C2 39                       RTS                RETURN.
1351                        *
1352                        *
1353 C7C3                   SINFACT  EQU   *
1354 C7C3 6E 38 EF 1D              FCB    $6E,$38,$EF,$1D      +(1/9!)
1355 C7C7 74 D0 0D 01              FCB    $74,$D0,$0D,$01      -(1/7!)
1356 C7CB 7A 08 88 89              FCB    $7A,$08,$88,$89      +(1/5!)
1357 C7CF 7E AA AA AB              FCB    $7E,$AA,$AA,$AB      -(1/3!)
1358                        *
1359                        *
1360 C7D3                   COSFACT  EQU   *
1361 C7D3 71 50 0D 01              FCB    $71,$50,$0D,$01      +(1/8!)
1362 C7D7 77 B6 0B 61              FCB    $77,$B6,$0B,$61      -(1/6!)
1363 C7DB 7C 2A AA AB              FCB    $7C,$2A,$AA,$AB      +(1/4!)
1364 C7DF 80 80 00 00              FCB    $80,$80,$00,$00      -(1/2!)
1365                        *
1366                        *
1367 C7E3 81 00 00 00       ONE    FCB    $81,$00,$00,$00      1.0
1368 C7E7 82 49 0F DB       PI     FCB    $82,$49,$0F,$DB      3.1415927
1369 C7EB 89 34 00 00       THREE60 FCB   $89,$34,$00,$00      360.0
1370                        *
1371                        *
1372                        *
```

```
                            TTL    FLTTAN
                    ********************************************************************************
                    *                                                                              *
                    *                          FLOATING POINT TANGENT                              *
                    *                                                                              *
                    ********************************************************************************
                    *
                    *
C7EF                FLTTAN  EQU    *
C7EF BD C8 39               JSR    PSHFPAC2     SAVE FPACC2 ON THE STACK.
C7F2 BD C6 46               JSR    TFR1TO2       PUT A COPY OF THE ANGLE IN FPACC2.
C7F5 BD C6 6F               JSR    FLTCOS        GET COSINE OF THE ANGLE.
C7F8 BD C7 AA               JSR    EXG1AND2      PUT RESULT IN FPACC2 & PUT ANGLE IN FPACC1.
C7FB BD C6 53               JSR    FLTSIN        GET SIN OF THE ANGLE.
C7FE BD C3 0A               JSR    FLTDIV        GET TANGENT OF ANGLE BY DOING SIN/COS.
C801 24 08                  BCC    FLTTAN1       IF CARRY CLEAR, ANSWER OK.
C803 CE C8 0F               LDX    #MAXNUM       TANGENT OF 90 WAS ATTEMPTED. PUT LARGEST
C806 BD C8 50               JSR    GETFPAC1      NUMBER IN FPACC1.
C809 86 07                  LDAA   #TAN90ERR     GET ERROR CODE IN A.
C80B BD C8 43       FLTTAN1 JSR    PULFPAC2      RESTORE FPACC2.
C80E 39                     RTS                  RETURN.
                    *
                    *
C80F                MAXNUM  EQU    *
C80F FE 7F FF FF            FCB    $FE,$7F,$FF,$FF        LARGEST POSITIVE NUMBER WE CAN HAVE.
                    *
                    *
                    *
```

```
                                TTL    TRIGUTIL
                        ********************************************************************************
                        *                                                                              *
                        *                              TRIG UTILITIES                                  *
                        *                                                                              *
                        *       The routines "DEG2RAD" and "RAD2DEG" are used to convert angles        *
                        *       from degrees-to-radians and radians-to-degrees respectively. The       *
                        *       routine "GETPI" will place the value of PI into FPACC1. This           *
                        *       routine should be used if the value of PI is needed in calculations    *
                        *       since it is accurate to the full 24-bits of the mantissa.              *
                        *                                                                              *
                        ********************************************************************************
                        *
                        *
C813                    DEG2RAD  EQU    *
C813 BD C8 39                    JSR    PSHFPAC2     SAVE FPACC2.
C816 CE C8 31                    LDX    #PIOV180     POINT TO CONVERSION CONSTANT PI/180.
C819 BD C8 66           DEG2RAD1 JSR    GETFPAC2     PUT IT INTO FPACC2.
C81C BD C1 93                    JSR    FLTMUL       CONVERT DEGREES TO RADIANS.
C81F BD C8 43                    JSR    PULFPAC2     RESTORE FPACC2.
C822 39                          RTS                 RETURN. (NOTE! DON'T REPLACE THE "JSR/RTS" WITH
                        *                            A "JMP" IT WILL NOT WORK.)
                        *
                        *
C823                    RAD2DEG  EQU    *
C823 BD C8 39                    JSR    PSHFPAC2     SAVE FPACC2.
C826 CE C8 35                    LDX    #C180OVPI    POINT TO CONVERSION CONSTANT 180/PI.
C829 20 EE                       BRA    DEG2RAD1     GO DO CONVERSION & RETURN.
                        *
                        *
C82B                    GETPI    EQU    *
C82B CE C7 E7                    LDX    #PI          POINT TO CONSTANT "PI".
C82E 7E C8 50                    JMP    GETFPAC1     PUT IT IN FPACC1 AND RETURN.
                        *
                        *
C831                    PIOV180  EQU    *
C831 7B 0E FA 35                 FCB    $7B,$0E,$FA,$35
                        *
C835                    C180OVPI EQU    *
C835 86 65 2E E1                 FCB    $86,$65,$2E,$E1
                        *
                        *
                        *
```

```
1444                            TTL    PSHPULFPAC2
1445                    ********************************************************************************
1446                    *                                                                              *
1447                    *       The following two subroutines, PSHFPAC2 & PULPFAC2, push FPACC2         *
1448                    *       onto and pull FPACC2 off of the hardware stack respectively.           *
1449                    *       The number is stored in the "memory format".                           *
1450                    *                                                                              *
1451                    ********************************************************************************
1452                    *
1453                    *
1454 C839               PSHFPAC2 EQU    *
1455 C839 38                    PULX                GET THE RETURN ADDRESS OFF OF THE STACK.
1456 C83A 3C                    PSHX                ALLOCATE FOUR BYTES OF STACK SPACE.
1457 C83B 3C                    PSHX
1458 C83C 8F                    XGDX                PUT THE RETURN ADDRESS IN D.
1459 C83D 30                    TSX                 POINT TO THE STORAGE AREA.
1460 C83E 37                    PSHB                PUT THE RETURN ADDRESS BACK ON THE STACK.
1461 C83F 36                    PSHA
1462 C840 7E C8 8C              JMP    PUTFPAC2     GO PUT FPACC2 ON THE STACK & RETURN.
1463                    *
1464                    *
1465 C843               PULFPAC2 EQU    *
1466 C843 30                    TSX                 POINT TO THE RETURN ADDRESS.
1467 C844 08                    INX                 POINT TO THE SAVED NUMBER.
1468 C845 08                    INX
1469 C846 BD C8 66              JSR    GETFPAC2     RESTORE FPACC2.
1470 C849 38                    PULX                GET THE RETURN ADDRESS OFF THE STACK.
1471 C84A 31                    INS                 REMOVE THE NUMBER FROM THE STACK.
1472 C84B 31                    INS
1473 C84C 31                    INS
1474 C84D 31                    INS
1475 C84E 6E 00                 JMP    0,X          RETURN.
1476                    *
1477                    *
1478                    *
```

```
                            TTL    GETFPAC
*******************************************************************************
*                                                                             *
*                          GETFPACx SUBROUTINE                                *
*                                                                             *
*       The GETFPAC1 and GETFPAC2 subroutines get a floating point number     *
*       stored in memory and put it into either FPACC1 or FPACC2 in a format  *
*       that is expected by all the floating point math routines. These       *
*       routines may easily be replaced to convert any binary floating point  *
*       format (i.e. IEEE format) to the format required by the math          *
*       routines.  The "memory" format converted by these routines is shown   *
*       below:                                                                *
*                                                                             *
*       31______24 23 22____________________0                                 *
*        exponent   s          mantissa                                       *
*                                                                             *
*       The exponent is biased by 128 to facilitate floating point            *
*       comparisons.  The sign bit is 0 for positive numbers and 1            *
*       for negative numbers.  The mantissa is stored in hidden bit           *
*       normalized format so that 24 bits of precision can be obtained.       *
*       Since a normalized floating point number always has its most          *
*       significant bit set, we can use the 24th bit to hold the mantissa     *
*       sign.  This allows us to get 24 bits of precision in the mantissa     *
*       and store the entire number in just 4 bytes.  The format required by  *
*       the math routines uses a seperate byte for the sign, therfore each    *
*       floating point accumulator requires five bytes.                       *
*                                                                             *
*******************************************************************************
*
*
C850                    GETFPAC1 EQU   *
C850 EC 00                       LDD   0,X          GET THE EXPONENT & HIGH BYTE OF THE MANTISSA,
C852 27 0B                       BEQ   GETFP12      IF NUMBER IS ZERO, SKIP SETTING THE MS BIT.
C854 7F 00 04                    CLR   MANTSGN1     SET UP FOR POSITIVE NUMBER.
C857 5D                          TSTB               IS NUMBER NEGATIVE?
C858 2A 03                       BPL   GETFP11      NO. LEAVE SIGN ALONE.
C85A 73 00 04                    COM   MANTSGN1     YES. SET SIGN TO NEGATIVE.
C85D CA 80              GETFP11  ORAB  #$80         RESTORE MOST SIGNIFICANT BIT IN MANTISSA.
C85F DD 00              GETFP12  STD   FPACC1EX     PUT IN FPACC1.
C861 EC 02                       LDD   2,X          GET LOW 16-BITS OF THE MANTISSA.
C863 DD 02                       STD   FPACC1MN+1   PUT IN FPACC1.
C865 39                          RTS                RETURN.
                        *
                        *
C866                    GETFPAC2 EQU   *
C866 EC 00                       LDD   0,X          GET THE EXPONENT & HIGH BYTE OF THE MANTISSA,
C868 27 0B                       BEQ   GETFP22      IF NUMBER IS 0, SKIP SETTING THE MS BIT.
C86A 7F 00 09                    CLR   MANTSGN2     SET UP FOR POSITIVE NUMBER.
C86D 5D                          TSTB               IS NUMBER NEGATIVE?
C86E 2A 03                       BPL   GETFP21      NO. LEAVE SIGN ALONE.
C870 73 00 09                    COM   MANTSGN2     YES. SET SIGN TO NEGATIVE.
C873 CA 80              GETFP21  ORAB  #$80         RESTORE MOST SIGNIFICANT BIT IN MANTISSA.
C875 DD 05              GETFP22  STD   FPACC2EX     PUT IN FPACC1.
C877 EC 02                       LDD   2,X          GET LOW 16-BITS OF THE MANTISSA.
C879 DD 07                       STD   FPACC2MN+1   PUT IN FPACC1.
C87B 39                          RTS                RETURN.
                        *
                        *
                        *
```

```
                                TTL    PUTFPAC
                        ********************************************************************************
                        *                                                                              *
                        *                          PUTFPACx SUBROUTINE                                 *
                        *                                                                              *
                        *       These two subroutines perform to opposite function of GETFPAC1 and     *
                        *       GETFPAC2. Again, these routines are used to convert from the           *
                        *       internal format used by the floating point package to a "memory"       *
                        *       format. See the GETFPAC1 and GETFPAC2, documentation for a              *
                        *       description of the "memory" format.                                    *
                        *                                                                              *
                        ********************************************************************************
                        *
                        *
C87C                    PUTFPAC1 EQU    *
C87C DC 00                       LDD    FPACC1EX      GET FPACC1 EXPONENT & UPPER 8 BITS OF MANT.
C87E 7D 00 04                    TST    MANTSGN1      IS THE NUMBER NEGATIVE?
C881 2B 02                       BMI    PUTFP11       YES.  LEAVE THE M.S. BIT SET.
C883 C4 7F                       ANDB   #$7F          NO.  CLEAR THE M.S. BIT.
C885 ED 00              PUTFP11  STD    0,X           SAVE IT IN MEMORY
C887 DC 02                       LDD    FPACC1MN+1    GET L.S. 16 BITS OF THE MANTISSA.
C889 ED 02                       STD    2,X
C88B 39                          RTS
                        *
                        *
C88C                    PUTFPAC2 EQU    *
C88C DC 05                       LDD    FPACC2EX      GET FPACC1 EXPONENT & UPPER 8 BITS OF MANT.
C88E 7D 00 09                    TST    MANTSGN2      IS THE NUMBER NEGATIVE?
C891 2B 02                       BMI    PUTFP21       YES.  LEAVE THE M.S. BIT SET.
C893 C4 7F                       ANDB   #$7F          NO.  CLEAR THE M.S. BIT.
C895 ED 00              PUTFP21  STD    0,X           SAVE IT IN MEMORY
C897 DC 07                       LDD    FPACC2MN+1    GET L.S. 16 BITS OF THE MANTISSA.
C899 ED 02                       STD    2,X
C89B 39                          RTS
                        *
```

```
ADDNXTD    C0D2 *0229 0145 0221
ADDNXTD1   C10E *0261 0235 0238 0246 0250 0258
ANGRED     C759 *1298 1144 1170
ANGRED1    C76D *1309 1307
ANGRED2    C773 *1311 1315
ANGRED3    C780 *1316 1312
ANGRED4    C792 *1324 1318 1320
ANGRED5    C7A7 *1332 1326
ASCFLT     C000 *0095
ASCFLT1    C014 *0105
ASCFLT10   C0BC *0212 0144
ASCFLT11   C0C1 *0218 0128 0225
ASCFLT12   C0B6 *0206 0204
ASCFLT13   C070 *0172 0167
ASCFLT14   C0AD *0202 0191
ASCFLT15   C082 *0181 0177
ASCFLT16   C085 *0182 0179
ASCFLT2    C01B *0111
ASCFLT3    C02A *0122 0112
ASCFLT4    C043 *0142 0107 0117 0147
ASCFLT5    C039 *0132 0123 0127 0180 0185 0201
ASCFLT6    C050 *0151 0155
ASCFLT7    C069 *0166 0157 0163 0213 0220
ASCFLT8    C05F *0161 0165 0223
ASCFLT9    C08D *0186 0175
C1800VPI   C835 *1439 1427
CHCK0      C180 *0331 0277 0315 0361 0364 0466 0472 0610 0616 0736
                    0961 1021 1075
CHCK01     C188 *0337 0335
CONST10    C18F *0342 0291 0772
CONSTP1    C18B *0341 0287 0777
CONSTP5    C549 *0919 0779
COSFACT    C7D3 *1360 1213
DECDIG     C52C *0903 0818
DEG2RAD    C813 *1415 1147 1173
DEG2RAD1   C819 *1418 1428
DIV0ERR    0004 *0041 0612
EXG1AND2   C7AA *1337 1217 1385
EXPSIGN    0000 *0089 0181 0203
FINISH     C117 *0274 0168 0208
FINISH1    C140 *0291 0286
FINISH2    C146 *0293 0290 0295
FINISH3    C14E *0296 0278 0285
FLT2INT    C5A1 *1019
FLT2INT1   C5CA *1038 1027
FLT2INT2   C5E3 *1052 1025
FLT2INT3   C5EA *1055 1022
FLT2INT4   C5DF *1049 1029 1039
FLT2INT5   C5BA *1031 1035 1041
FLTADD     C23C *0463 0582 0781 1102 1275 1289 1308 1323 1330
FLTADD1    C24C *0471 0467
FLTADD10   C248 *0469 0543 0560
FLTADD11   C2DC *0546 0504
FLTADD12   C2D9 *0544 0540 0552 0557
FLTADD2    C262 *0481 0473
FLTADD3    C27B *0494 0485
FLTADD4    C254 *0474 0488
FLTADD5    C282 *0497 0493 0501
FLTADD6    C247 *0468 0480 0495 0544
FLTADD7    C28B *0502 0483
FLTADD8    C2AA *0519 0506
FLTADD9    C2CE *0539 0525
FLTASC     C3DB *0733
FLTASC1    C3EB *0742 0737
FLTASC10   C49B *0821 0854
FLTASC11   C49E *0822 0830
```

```
TFR1TO2    C646 *1123 1091 1104 1209 1215 1383
THREE60    C7EB *1369 1304
TOLGSMER   0005 *0042 1049
UINT2FLT   C577 *0959 0992
UINTFLT1   C580 *0964 0962
UMULT      C1D9 *0395 0387
UMULT1     C1E3 *0402 0418
UMULT2     C1F4 *0411 0404
UMULT3     C217 *0426 0420
UMULT4     C234 *0439 0427 0434
UNFERR     0003 *0040 0382 0541 0664
```

PAGE OUT OF SEQUENCE

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands Milton Keynes, MK145BP, England.
HONG KONG: Motorola Inc.; International Semiconductor Group; P.O. Box 80300; Cheung Sha Wan Post Office; Kowloon Hong Kong.

MOTOROLA

A21956 PRINTED IN USA 9-87 IMPERIAL LITHO 50748 18,000 AN974

```
FLTASC12   C4B1 *0831 0828
FLTASC13   C4E2 *0856 0859
FLTASC14   C500 *0873 0869
FLTASC15   C519 *0888 0862
FLTASC16   C4D3 *0848 0844
FLTASC17   C45C *0791 0786
FLTASC18   C490 *0817 0813
FLTASC2    C40E *0760 0757
FLTASC3    C438 *0776 0766
FLTASC4    C440 *0779 0770
FLTASC5    C414 *0763 0775
FLTASC6    C430 *0773 0778
FLTASC7    C452 *0787 0791
FLTASC8    C474 *0804 0797 0799
FLTASC9    C493 *0818 0807
FLTCMP     C54D *0923 0765 0769 1311 1325
FLTCMP1    C563 *0933 0930 0936 0939
FLTCMP2    C564 *0934 0925 0927
FLTCOS     C66F *1168 1384
FLTCOS1    C68C *1183 1179 1181
FLTDIV     C30A *0608 1101 1387
FLTDIV1    C316 *0615 0611
FLTDIV10   C399 *0684 0679
FLTDIV11   C3C1 *0703 0700
FLTDIV12   C3D7 *0712 0708
FLTDIV13   C3C4 *0704 0702
FLTDIV14   C35E *0655 0639
FLTDIV2    C320 *0620 0617
FLTDIV3    C33E *0636 0633
FLTDIV4    C350 *0648 0636
FLTDIV5    C375 *0666 0662
FLTDIV6    C348 *0642 0665 0713
FLTDIV7    C377 *0667 0663
FLTDIV8    C345 *0640 0666
FLTDIV9    C37B *0669 0691
FLTFMTER   0001 *0038 0136
FLTMUL     C193 *0358 0293 0774 1210 1218 1245 1419
FLTSIN     C653 *1142 1386
FLTSIN1    C65F *1149
FLTSIN2    C66A *1154 1152 1183
FLTSQR     C5EB *1073
FLTSQR1    C5F4 *1078 1076
FLTSQR2    C5FD *1083 1079
FLTSQR3    C617 *1098 1095
FLTSQR4    C61A *1100 1097
FLTSQR5    C61C *1101 1110
FLTSUB     C2FE *0580 1313
FLTSUB1    C303 *0583 0581
FLTTAN     C7EF *1381
FLTTAN1    C80B *1392 1388
FPACC1EX   0000 *0030 0100 0101 0151 0206 0207 0224 0276 0279 0282
                      0314 0320 0360 0368 0374 0386 0388 0425 0438 0471 0475
                      0481 0490 0491 0556 0615 0658 0668 0735 0742 0782 0898
                      0929 0937 0960 0965 1020 1023 1036 1053 1074 1088 1103
                      1106 1112 1124 1199 1221 1242 1249 1338 1341 1517 1553
FPACC1MN   0001 *0031 0229 0231 0234 0237 0241 0244 0248 0249 0254
                      0255 0256 0259 0262 0264 0317 0319 0323 0325 0369 0405
                      0408 0441 0443 0477 0492 0509 0512 0515 0518 0519 0521
                      0522 0524 0526 0529 0533 0536 0546 0548 0549 0551 0553
                      0554 0555 0631 0634 0648 0650 0651 0653 0669 0671 0673
                      0675 0676 0678 0681 0683 0687 0688 0689 0695 0697 0704
                      0707 0709 0710 0711 0744 0787 0788 0789 0822 0824 0825
                      0827 0831 0833 0834 0836 0896 0932 0940 0985 0991 1031
                      1032 1033 1042 1046 1054 1086 1108 1114 1126 1197 1219
                      1244 1247 1342 1345 1519 1558
FPACC2EX   0005 *0033 0363 0375 0465 0474 0482 0484 0609 0637 0659
```

```
                    0928 0938 1092 1100 1111 1125 1272 1286 1309 1310 1324
                    1327 1339 1340 1531 1564
FPACC2MN    0006 *0034 0230 0232 0239 0243 0261 0263 0402 0414 0415
                    0416 0421 0426 0476 0496 0507 0510 0513 0516 0520 0523
                    0547 0550 0632 0635 0649 0652 0655 0656 0657 0674 0677
                    0696 0698 0931 0941 1113 1127 1274 1288 1343 1344 1533
                    1569
FPMULT1     C1C1 *0381 0376
FPMULT2     C1C8 *0385 0377 0381
FPMULT3     C1CF *0388 0362 0370
FPMULT4     C1AE *0371 0365
FPMULT5     C1BC *0378 0389
FPMULT6     C1D5 *0391 0380 0384
FPNORM      C161 *0313 0283 0539 0966
FPNORM1     C16D *0319
FPNORM2     C16F *0320 0324
FPNORM3     C17C *0326 0316 0318
FPNORM4     C17E *0328 0321
GETFP11     C85D *1516 1514
GETFP12     C85F *1517 1511
GETFP21     C873 *1530 1528
GETFP22     C875 *1531 1525
GETFPAC1    C850 *1509 1390 1433
GETFPAC2    C866 *1523 0288 0292 0764 0768 0773 0780 1232 1281 1305
                    1418 1469
GETPI       C82B *1431
MANTSGN1    0004 *0032 0102 0113 0371 0373 0479 0502 0505 0538 0622
                    0623 0746 0756 0758 0894 0924 0934 0997 1026 1047 1078
                    1128 1153 1182 1201 1223 1240 1251 1306 1322 1329 1346
                    1349 1512 1515 1554
MANTSGN2    0009 *0035 0372 0478 0503 0583 0585 0621 0926 0935 1129
                    1270 1284 1347 1348 1526 1529 1565
MAXNUM      C80F *1396 1389
N9999999    C545 *0916 0763
NSQRTERR    0006 *0043 1080
NUMERIC     C155 *0303 0106 0116 0126 0143 0154 0162 0174 0184 0190
                    0219
NUMERIC1    C15F *0310 0305 0307
ONE         C7E3 *1367 1280
OVFERR      0002 *0039 0378 0558 0640
P9999999    C541 *0913 0767
PI          C7E7 *1368 1432
PIOV180     C831 *1436 1417
PSHFPAC2    C839 *1454 0097 0359 0464 0620 0748 1083 1143 1169 1382
                    1416 1426
PULFPAC2    C843 *1465 0134 0298 0391 0469 0646 0892 1118 1155 1392
                    1420
PUTFP11     C885 *1557 1555
PUTFP21     C895 *1568 1566
PUTFPAC1    C87C *1552
PUTFPAC2    C88C *1563 1462
PWR10EXP    0001 *0090 0187 0192 0195 0199 0202 0280 0284 0289 0294
RAD2DEG     C823 *1425
SINCOS      C68F *1195 1149 1176
SINCOS1     C6BC *1218 1226
SINCOS2     C6D3 *1230 1259
SINCOS3     C713 *1262 1277
SINCOS4     C6BF *1219 1216
SINCOS5     C73F *1283 1279
SINCOS6     C74E *1289 1282
SINCOS7     C6B9 *1217 1212
SINFACT     C7C3 *1353 1203
SINT2FLT    C589 *0984
SINTFLT1    C595 *0992 0987
SINTFLT2    C59F *0998 0996
TAN90ERR    0007 *0044 1391
```